FR1 2

# A NETWORK OF CONCURRENT MICROCOMPUTERS FOR CLOSED LOOP INTERACTIONS

BY

DOUGLAS LEONARD BRAY

B.S., Clarkson College of Technology, 1981

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1982

Urbana, Illinois

# ACKNOWLEDGEMENTS

I wish there were a more adequate way to thank my thesis advisor, Professor Ricardo Uribe, for his inspiration, guidance and assistance, without whom this project would not have even been started. For their support and aid throughout the project, I would like to thank Jeff Konicek and Bill Santic. Special thanks must go to Kevin Warren for supplying hardware and software for the demonstration, and to Martin Eberhard for the hardware he made available to complete the demonstration.

# PREFACE

The following thesis is one of the leading contributions in microcomputer network design from a cybernetic viewpoint. This novel research transcends the conventional, "efficient" use of microcomputers and explores the network of microcomputers per se as the new entity. Concepts such as organizational closure and autopoiesis are hinted at from this perspective. Cognition can now be studied under a new light with a set of concurrent microcomputers that approaches the structure and function of a nervous system.

Furthermore, the present thesis shows how a set of independent microcomputers can both: interact among each other and, as a network, interact with its environment.

Although the use of a multiprocessor architecture in the field of robotics and elsewhere is not new, the network of microcomputers presented here offers an illuminating answer to the problem of handling the increasing number of microcomputers involved, as well as a new, radical approach to robotics and other fields of enormous potential.

Ricardo B. Uribe
University of Illinois
Urbana, 1982

TABLE OF CONTENTS

1. INTRODUCTION ..... 1

2. SYSTEM OVERVIEW ..... 4

2.1 Project Goals ..... 4

2.2 The Asynchronous Protocol for Interaction ..... 5

2.3 Overview of the Hardware ..... 10

2.4 Overview of the Software ..... 11

3. SYSTEM HARDWARE ..... 14

3.1 The Microcomputer Node ..... 14

3.2 The Microcomputer Node Circuit Boards ..... 31

3.3 Bus Topology and Backplane ..... 37

3.4 Terminal Interface and Master Reset Board ..... 37

4. SYSTEM SOFTWARE ..... 43

4.1 Invoking the Monitor ..... 44

4.2 The List of Monitor Commands ..... 44

4.3 List of Utility Routines ..... 46

4.4 Requirements of the User Programs ..... 48

5. A DEMONSTRATION ..... 50

6. CONCLUDING REMARKS ..... 57

APPENDIX A -- MONITOR LISTING ..... 59

APPENDIX B -- DEMONSTRATION SOFTWARE LISTINGS ..... 74

APPENDIX C -- INSTRUCTIONS FOR PRINTED CIRUIT BOARD CONSTRUCTION .. 90

APPENDIX D -- INSTRUCTIONS FOR ASSEMBLING, LINKING
AND PROGRAMMING THE 8751 EPROM ..... 99

REFERENCES ..... 104

# LIST OF FIGURES

2.1 Simplified Block Diagram of
R. Uribe's Network Interaction Scheme ..... 6
2.2 Block Diagram of a Node's Interaction Hardware ..... 8
2.3 Inter-node Interaction Block Diagram ..... 9
2.4 System Hardware Block Diagram ..... 12
3.1 Node Schematic with the 8751, XRAM and Buffers ..... 15
3.2 Node Schematic with the Buffer Control Logic ..... 16
3.3 Algorithm for the Buffer Control Logic ..... 23
3.4 I/O Timing Diagram ..... 27
3.5 An Example of an I/O Device ..... 29
3.6 Printed Circuit Board Chip Layout ..... 33
3.7 BCL Schematic with Pin Numbers for Printed Circuit Board ..... 34
3.8 Chip Layout for Prototype Board ..... 35
3.9 BCL Schematic with Pin Numbers for Prototype ..... 36
3.10 3-Dimensional Bus Structure ..... 38
3.11 Bus Wiring Diagram ..... 39
3.12 Bus Pin Definition ..... 40
3.13 Terminal Board Schematic ..... 42
5.1 Demonstration Hardware Setup ..... 52
5.2 Node Interaction for Demonstration ..... 54
C.1 Front Side of Printed Circuit Board ..... 93
C.2 Back Side of Printed Circuit Board ..... 94

# LIST OF TABLES

3.1 BCL Signal Description ..... 22

C.1 Parts List for One Node Board ..... 92

# CHAPTER 1

## INTRODUCTION

Webster's New International Dictionary defines the word network as "Any system of lines or channels interacting or crossing like fabric of a net." When individual and independent threads are woven into a net, the importance of the individual threads is lost and the net they now form becomes important. If one thread breaks, the net may have a hole but it is still a net and it holds as such.

Like the threads in the net, when microcomputers, individual and independent, join together and interlace, a new entity emerges, a microcomputer network, that did not exist before and they too lose their relevance as individual microcomputers. It is this new entity, the microcomputer network, that now becomes relevant.

The network of microcomputers described in the following chapters has been designed and built to provide a tool with which to investigate

how these entities interact with their environment, that may include other entities (e.g., people), and with themselves.

At the core of the notion of life, cognition and perception lies the concept of organizational closure. Systems are said to be organizationally closed, and called system-wholes if, as explained by Franciso J. Varela (4), "their organization is a circular network of interactions rather than a tree of hierarchical processes". It is this concept of organizational closure that we had in mind when the network was designed, and it is this same approach that should be taken when the network is further studied.

This concept implies closed loops, and so does any interaction of the network with its environment, other similar entities, or even itself. As an example of these loops let us consider human beings reaching for an object. First, they see the distance between the object and their hand. Their brain then tells their hand and arm to reduce the distance (or error with respect to the goal). This generates a new distance to be seen by their eyes, thus closing the loop. This new distance represents a new, smaller, error. The loop continues to reduce the error until the hand has reached the object.

Memory of past events is also a consequence of closed loops in the nervous system that reactivates the same patterns of activity that originated the remembrance. A similar situation can be realized in the network of microcomputers as if it were a network of neurons, in which

closed loops of activity can be generated and reactivated.

The following chapters describe the network and the software required by the network. They give instructions to future students as to how the system can be expanded and used. An example is also given to illustrate how the network can be used to explore interactions between different entities.

This thesis has been written such that it may be read and understood by a variety of students with different backgrounds and experiences.

# CHAPTER 2

# SYSTEM OVERVIEW

## 2.1 Project Goals

In order to provide an expandable tool for exploring the interactions between a network of microcomputers and its environment and similar entities, a modular microcomputer network in which the microcomputer nodes interact via an asynchronous protocol, had to be developed.

The network must be easy to expand to a large number of microcomputer nodes, and software must be provided to simplify the writing and debugging of user programs.

## 2.2 The Asynchronous Protocol for Interaction

In order to begin, an asynchronous protocol was needed. Professor Ricardo Uribe, in a paper soon to be published, describes the development of just such an interaction scheme. In his system each microcomputer contains a random access memory (RAM) that can be loaded by any microcomputer in the network and read by the microcomputer to whom the RAM belongs.

This allows a very simple interaction scheme. When a processor wants to act on another, it writes a message in a "mailbox" that the other microcomputer can read at its own discretion. The "mailbox" is simply a location in this RAM. The recipient microcomputer can act in a similar fashion on the microcomputer that initiated the interaction, thereby closing the interactive loop, that can go on and on. Other, more complex schemes can also be developed using this RAM. For example, a "program" running in the RAM could be altered by other nodes, thus changing the function of the microcomputer.

In Uribe's network, all microcomputers are tied to a single linear bus. When one processor wishes to talk to another it must capture the entire bus, disabling writes to the bus by the rest of the processors, and it writes to a RAM location in all processors that do not happen to be using its own RAM. When an individual processor wants to read its own RAM it blocks any incoming messages so they will not interfere with its reading. A block diagram of this system is shown in Fig. 2.1.

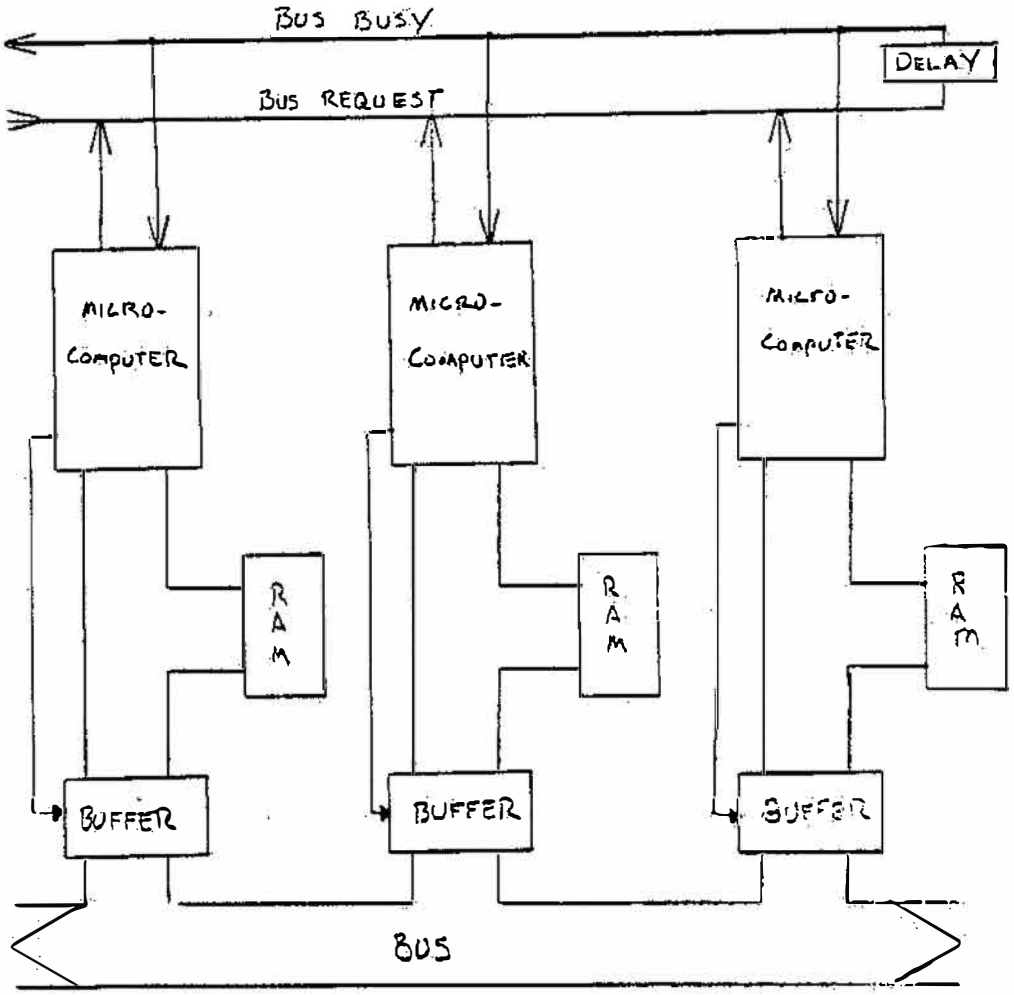



FIGURE 2.1

Simplified Block Diagram of R. Uribe's

Network Interaction Scheme

Although it was decided to continue with this mailbox protocol in the present project, the protocol has been taken a few steps further. Due to the potentially large number of microcomputers (nodes) in this network there was some concern that a simple linear bus would not be sufficient to handle the large number and frequency of inter-node interactions. Therefore, in this system, a three dimensional bus topology is used. In addition, each node has the ability to cut or "block" the bus (PBUS) at either side of its access point thereby defining a "region" of the network. The bus requests and captures also becomes slightly more complex, but far more interesting. When a processor wants to write a message to another node, it requests and captures its "section" or access point. This in turn requests the sections of bus on both sides of the node. If captured, the next sections are requested and the progression spreads "like wildfire" until all or as much as possible has been captured and the message is written to all non-busy RAM's along the way. The block diagram of the microcomputer node's inter-node interaction hardware can be seen in Fig. 2.2. In Fig. 2.3 the way in which the nodes are connected has been illustrated.

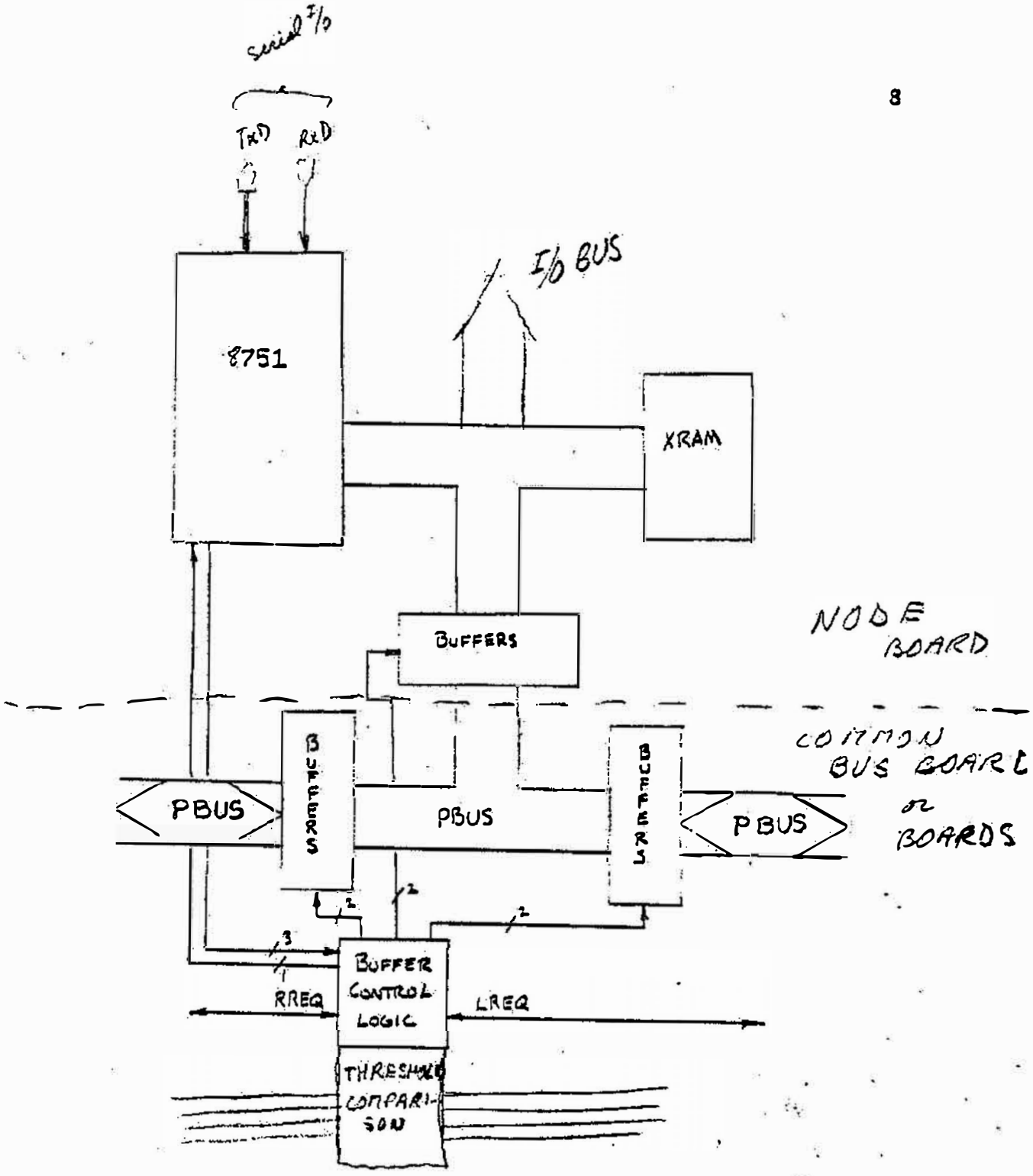



FIGURE 2.2

Block Diagram of a Node's Interaction Hardware

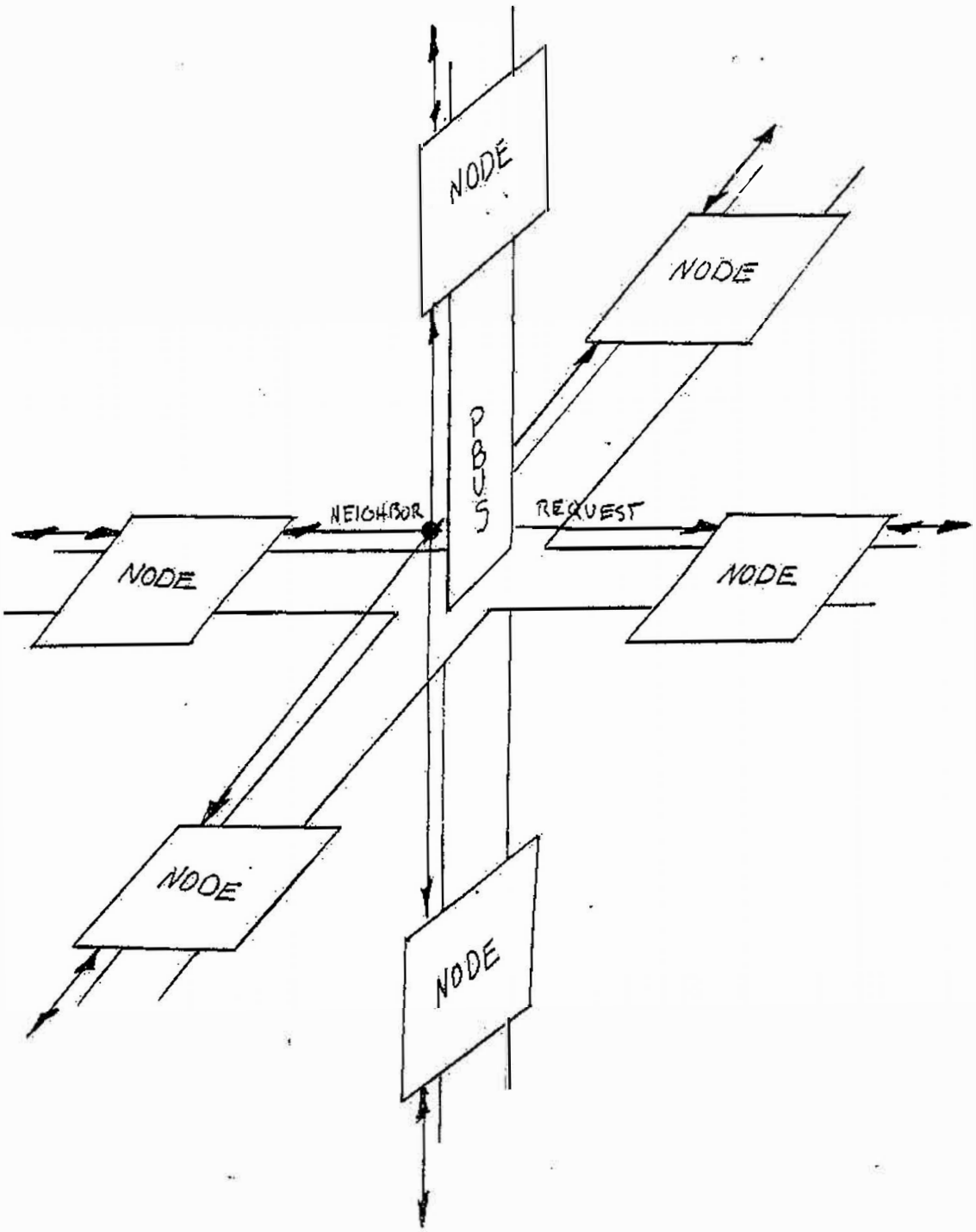



FIGURE 2.3

Inter-node Interaction Block Diagram

## 2.3 Overview of the hardware

The microcomputer nodes make up the major part of the system. Each microcomputer resides on a separate circuit board, all indentical. The microcomputer is based on the Intel 8751 microcontroller on a chip. This microcomputer chip is connected to the system via its address-data bus and its internal serial port.

In order to provide the mailbox locations, a 1024-byte random access memory is connected to the 8751 address-data bus and through bi-directional buffers it is connected to the node interaction bus (PBUS). Although the primary role of this RAM is to be the mailbox, some limited data storage and program execution can be performed in it. The buffers connect the RAM and the 8751 to the PBUS and their control logic is responsible for receiving and granting bus requests.

The 8751 also uses the address-data bus to interact with the environment through parallel memory mapped I/O devices. These I/O devices can make use of the 8751 pins that act as either port pins or as either interrupt or event counter inputs.

In order to allow the system and user software to interact with a terminal, the internal serial ports of all 8751s are connected to a simple linear bus (SBUS) which terminates at an RS-232 interface card. This card provides the TTL/RS-232 conversion and a clock to generate the baud rate that is required by the 8751 to talk to an RS-232 terminal.

The serial bus has a simple bus request/bus busy protocol for bus arbitration. This interface card also contains a reset circuit which feeds all cards with a master reset signal.

In order to make the network easily expandable to a large number of microcomputer nodes, the individual node boards must be easy to build. To achieve this, the number of components per card was kept small and a printed circuit board was made. This should allow someone to get a node built and running after only a few hours of work.

The block diagram in Fig. 2.4 shows the PBUS used for inter-node interaction and the SBUS with the terminal interface board including the RS-232 converter and master reset circuit.

## 2.4 Overview of the Software

The network software is divided into two portions: system software and user software.

### 2.4.1 The System Software

The system software is provided by a monitor program which is to reside in each node. The monitor includes some simple commands to aid in program debugging and execution. The monitor also includes many

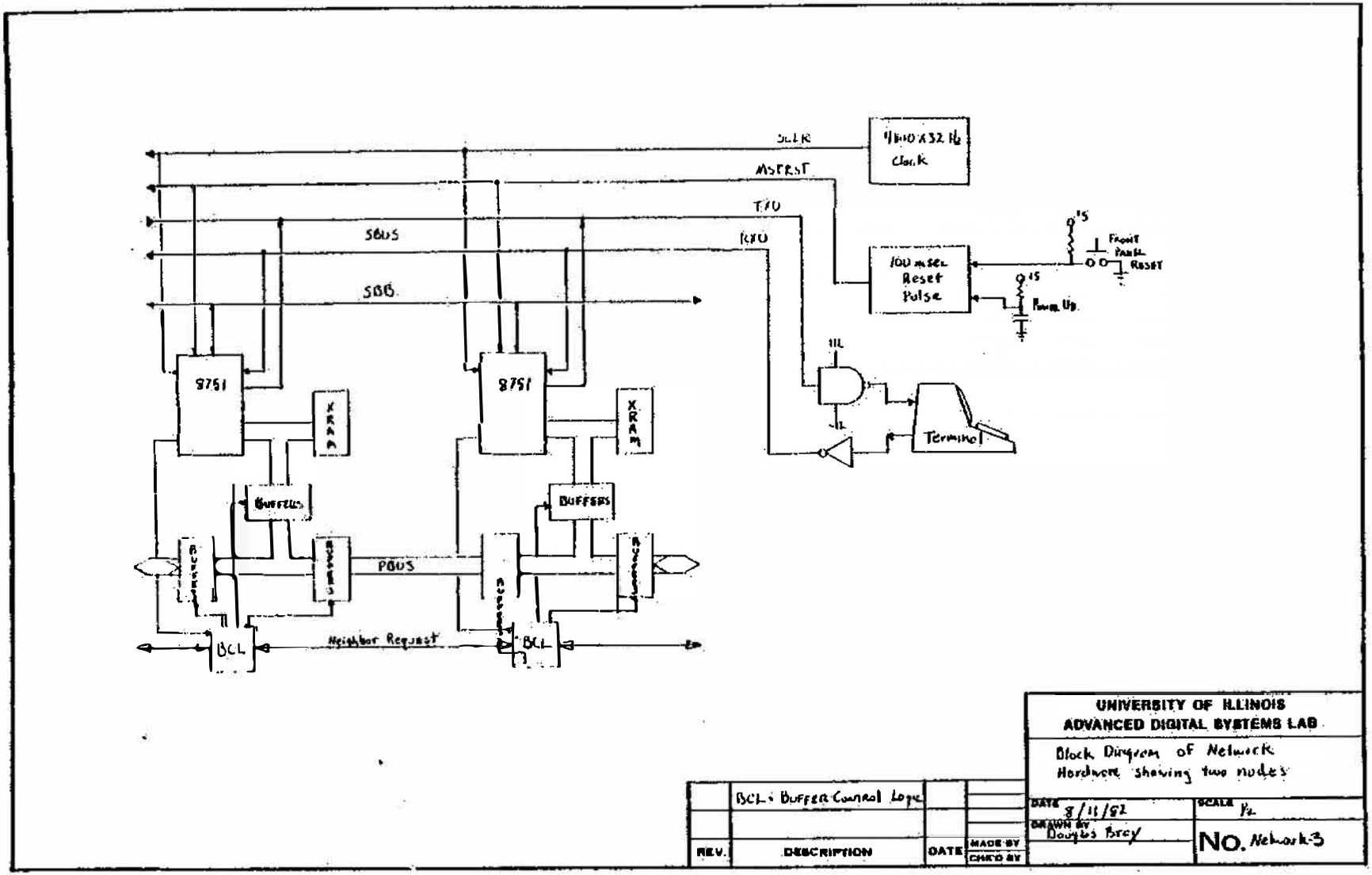



FIGURE 2.4

System Hardware Block Diagram

utility routines that may be called by user programs. These routines perform common operations such as reading and writing messages, I/O functions and input and output to the user terminal.

### 2.4.2 The User Software

The user software is also resident in each 8751's EPROM. The linker (discussed in Appendix D) is used to combine the monitor program and the user software. This allows users to access routines within the monitor and allows all the software to be programmed into the EPROMs as one unit.

As we see the future of the network, the user software will be of two classes, i) software for nodes "at the periphery" of the network that interact with the network's environment and ii) software for nodes "inside" the network that only interact with other nodes. Each node that interacts with the environment, must do so through transducers and therefore, the software will be determined by the particular transducer. On the other hand, software for "inside" nodes will be the same, since the interactions in this case are only with other identical nodes.

Moreover, we foresee that nodes will be able to develop different "programs" in their own RAMs that will reflect the activities of the network after it has been initially programmed by the user.

# CHAPTER 3

# SYSTEM HARDWARE

## 3.1 The Microcomputer Node

Each microcomputer node consists of a microcomputer on a chip, an external random access memory for inter-node interaction, a clock and reset circuitry, buffers and buffer control logic (which receives and regulates the bus requests), serial bus request logic and the I/O bus. The schematics for the node are in Figs. 3.1 and 3.2. For pin numbers for the buffer control logic refer to Figs. 3.7 and 3.9.

### 3.1.1 The Microcomputer on a Chip

The microcomputer used in this system is Intel's 8751 microcontroller on a single integrated circuit. This chip not only contains the central processing unit but contains a 4048 x 8-bit programmable read-only program store, 128 bytes of random access memory

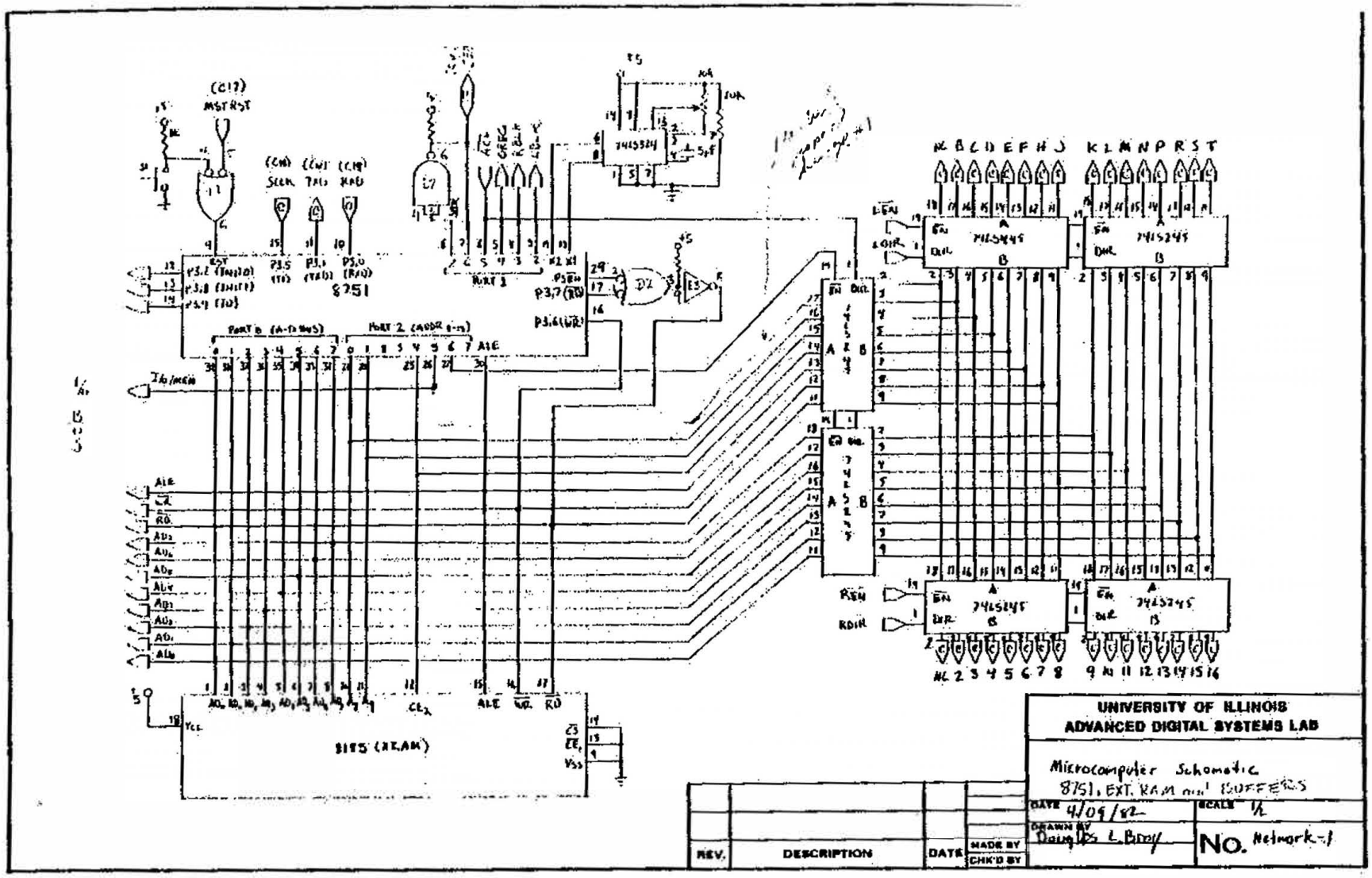



FIGURE 3.1

Node Schematic with the 8751, XRAM and Buffers

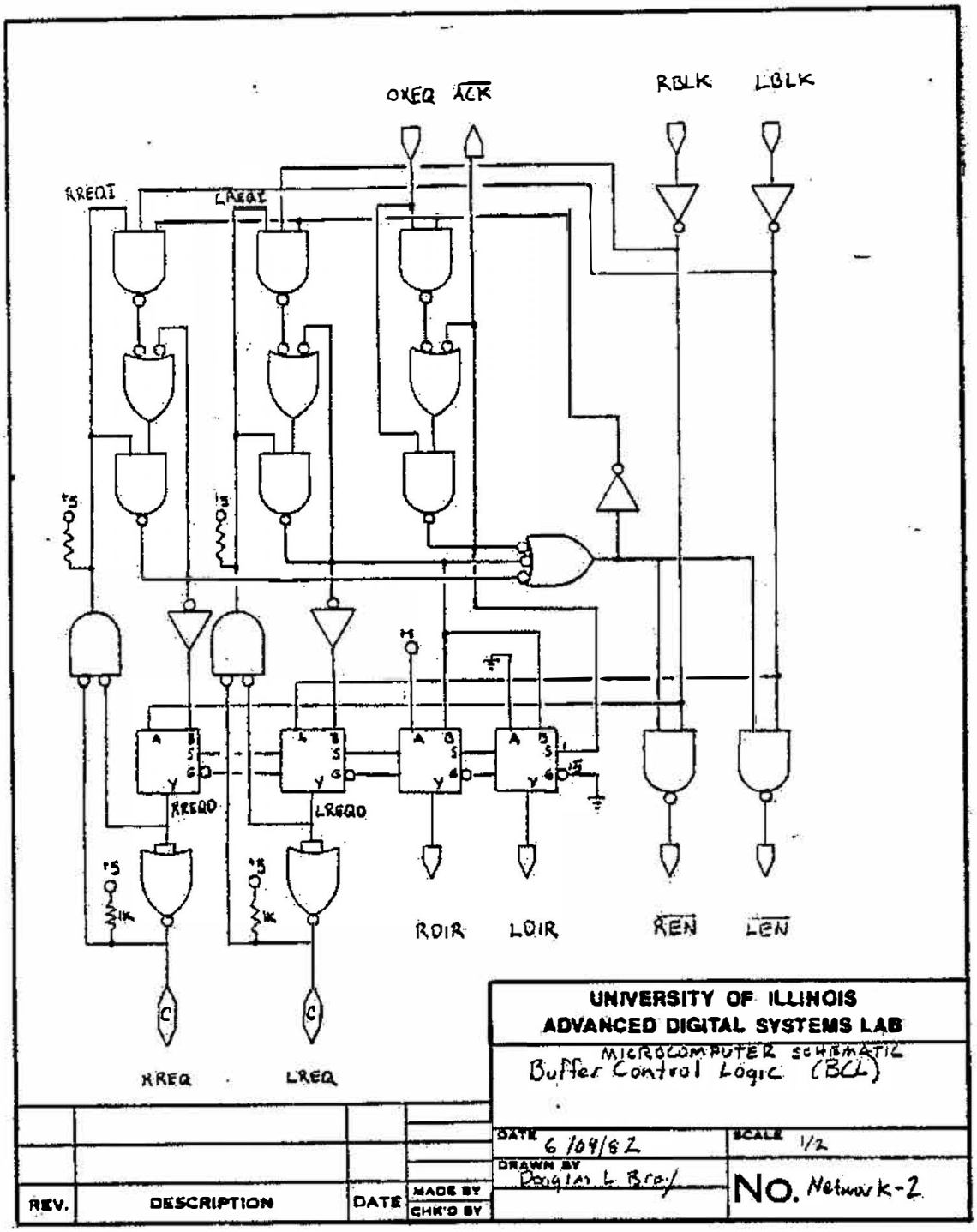



FIGURE 3.2

Node Schematic with the Buffer Control Logic

used for data store, 32 I/O lines divided into three 8-bit bi-directional ports, two 16-bit timer/counters, a serial I/O channel, a multi-level interrupt processor and a special boolean processor.

In addition to the internal memory, up to 60 kilobytes of external program memory and up to 64 kilobytes of external data memory can be accessed. Within the internal memory, 256 bits are individually addressable. These bits include bits within both special purpose registers and general purpose data bytes. These bits can be manipulated with the boolean processor, to perform operations such as jump if bit is set, clear bit, etc.

The I/O port pin's mode (input or output) is individually software programmable. These ports are unusual since there is no register to store the pin's mode. The mode is determined by the last value written to the port pin. If a pin is to act as an input, a "one" must first be written to the corresponding bit. The value applied to the pin may then be read. A pin being used as an output may simply have the data ("zero or one") written to that pin's bit.

### 3.1.2 Clock and Reset Circuit

Due to the asynchronous nature and required independance of all nodes within the network, each processor must have a separate clock. In order to avoid dead locks while attempting to capture the buses, it was

decided that each clock would run at a slighty different speed. To achieve this, each node has a voltage-controlled oscillator with trim pots to set the frequency. Since the available version of the 8751 runs at a maximum frequency of 8 megahertz, the VCO's are set for a frequency between 7 and 8 megahertz. A 74LS324 with an external capacitor of five picofarads was used, requiring approximately 2 volts at the 74LS324 pins 2 and 13, to give the desired frequency.

From past experience with networks in the Advanced Digital Systems Laboratory, it was decided that a master reset signal should be bused to all cards, in addition to the individual resets on each card. A simple push button on each card was logical ORed with the output of a monostable multivibrator that provides a reset pulse on either power up or when the front panel reset button is depressed.

### 3.1.3 External Random Access Memory

As described in the system overview, the external RAM's primary role is to allow the interactions among nodes. However, it was desirable to allow a limited amount of data storage and program execution within the RAM. To meet all these objectives, the RAM must be connected to the PBUS which will allow it to be connected to any processor's address-data bus (Port 0), some address lines from Port 2 and control lines, as well as to its own processor's lines. The RAM used is Intel's 8185, i.e., 1024 x 8-bit RAM with on chip address/data demultiplexer, which is

required by the 8751. The chip enable is connected to Port 2 pin 4 (address bit 12). This places the 1-kilobyte block of RAM in a variety of address spaces. The address spaces used by this system are 1000H-13FFH and 5000H-53FFH. The first range is used for the inter-node interaction and the second range's purpose will be explained in section 3.1.4.

The 8185's write signal (WR') is simply connected to the 8751's WR' line. When this line goes low it signals the RAM that valid data is on the address-data bus and a write is to be performed. The read signal is a little more complex. Since this external RAM is to be used as both data and program store, and the 8751 has separate read signals for external data memory (RD') and program memory (PSEN'), these two signals must be logically ORed. All of these lines are low active, so an AND gate is required. A NAND with an inverter at the output is used. This circuit can be seen in Fig. 3.1.

### 3.1.4 Buffers and Buffer Control Logic

Each microcomputer node contains three sets of bi-directional 3-state buffers. Each set is made up of two 74LS245s, which are 8-bits each. The first set allows the processor to cut itself and its own external RAM off the PBUS. This is required when performing reads on its external RAM, accessing the I/O or running code from the external RAM. During any of these "private" activities, the buffers are disabled

to prevent any other processor from writing to the RAM and interfering with the process. It can now be seen that only limited data storage and program execution may be done in the external RAM, the amount of time spent on private activities should be kept to a minimum, since during that time the buffers are disabled and no messages can get into that RAM and therefore will be lost. To control this enable, address bit 14 was used. A set on this bit makes any external (external to the 8751) access, either memory or I/O, "private", by disabling the buffers (they go into their high impedance state). Combining this with the address requirement of the last section, we see that 1000H-13FFH are for message writes since the buffers are enabled while the microcomputer uses 5000H-53FFH for reads and writes of only its own external RAM. The I/O addresses will be discussed in the next section.

The direction of the buffer is determined by a line that returns from buffer control logic (BCL) called ACK'. ACK'=0 signifies that the PBUS has been captured. The normal direction of this set of buffers must be toward the XRAM so it can "listen" for messages. It is important that a node's buffers only drive data on the bus when that node has captured the bus. Thus, ACK' is used to turn the buffers toward the bus when the bus has been captured and the processor is ready to write to the PBUS.

The other two sets of buffers are in the PBUS itself, one on either side of where the first set meets the PBUS. As discussed in the system overview, these buffers allow the network to be cut (the buffers are

blocked), thus splitting the network into regions. Within the individual regions, completely independent activities can take place. These buffers and their control logic cause the "wildfire spreading" of the bus requests and grants.

The logic to control the enables and directions of this set of buffers is derived from discrete logic. The schematic for it appears in Fig. 3.2. Table 3.1 shows the names for the input and outputs of the logic and their use. The algorithms in Fig. 3.3 aid in explaining the logic's function.

When a request for the "section" of bus comes in, either from the processor (OREQ) or from a neighboring node, the BCL will grant the request if and only if no one has the section currently, by enabling the unblocked buffers and setting them to the appropriate direction. Looking at the schematic (Fig. 3.2), the top row of NAND gates 'ANDs' the requests with the preconditions for request grant. For the OREQ, the only precondition is that no other requests have been granted. The neighbor requests have another precondition as well: the node must not have that side blocked, for region splitting or some other reason. This prevents requests from crossing region boundaries.

Once a set of preconditions is met, the next two gates latch the grant until the request goes inactive. This holds the grant even though the first preconditon will change since the request was granted. The rest of the circuit is quite straight forward, the three grants are ORed

TABLE 3.1

BCL Signal Description

| SIGNAL NAME | DESCRIPTION |
|---|---|
| OREQ | Open Request - processor PBUS request |
| LREQI | Left Request Input - PBUS request from left side |
| RREQI | Right Request Input - PBUS request from right side |
| ACK' | Acknowledge Bar - OREQ grant acknowledge |
| LBLK | Left Block - output from 8751 to block left side |
| RBLK | Right Block - output from 8751 to block right side |
| LREQO | Left Request Output - request out to left side |
| RREQO | Right Request Output - request out to right side |
| LDIR | Left Direction - control left buffer direction |
| RDIR | Right Direction - control right buffer direction |
| LEN' | Left Enable Bar - control left buffer direction |
| REN' | Right Enable Bar - control right buffer direction |

```
While either OREQ or either neighbor request (NREQ) are inactive:
        ACK´ = HIGH
        LEN´,REN´ = HIGH
        LREQ´,RREQ´ = HIGH
        LDIR,RDIR = HIGH

When the processor wants the PBUS:
  while OREQ is HIGH
    wait NREQ is not granted
      then ACK´ = LOW
           LDIR = LOW (point left)
           RDIR = HIGH (point right)
           if LBLK is LOW LEN´,LREQ´ = LOW
           if RBLK is LOW REN´,RREQ´ = LOW

When a neighboring node wants to send a message throught:
  while NREQ´ is LOW (N = L or R)
    if another request is not being granted
      AND NBLK is low
        then LDIR,RDIR = LOW if N=L, HIGH if N=R
             NEN´ = LOW
             if N´BLK is LOW (N´=R if N=L ect.)
               then (N´EN)´,(N´REQ)´ = LOW
```

FIGURE 3.3

ALGORITHM FOR THE BUFFER CONTROL LOGIC

together to produce the signal which signifies that a request was granted and that any other incoming requests will be blocked. The ORed grants are also fed through NAND gates with the block lines from the processor to enable the unblocked buffers. During this, two of the multiplexers of the 74LS157 produce the appropriate direction signals depending on which request was granted. The other two multiplexers on that chip generate the requests to the proper neighboring nodes. If the OREQ was granted (ACK´=0), requests must go out to the neighboring nodes on each side, provided that the side is not blocked. To accomplish this, the inverted version of the block signals are routed to each side, thus generating requests only to unblocked sides. If a neighbor request was granted, the request is routed to the opposite side from where the request came. After leaving the multiplexer, the requests are passed to the bi-directional neighbor request line via open collector NORs configured as NOTs. The requests are received from these lines through a NOR gate with one input from the multiplexer output. This prevents the node from being requested by itself, which would of course hang up the BCL.

Note that if the bus is busy while the processor is requesting its section of bus, the processor can wait until the bus is not busy and ACK´ goes low. This way it is insured to get the bus eventually. But since there is no bus grant acknowledgement back to the neighboring nodes, the node requesting a neighbor´s bus section will not know if it was captured. So it must simply request and perform the write and the message will get as far as it can before it encounters a section of bus

already being used by another processor.

Moreover, while a node is performing a private operation, an incoming message is passed along the PBUS and this particular node will not receive it. Hence, there is no guarantee that a particular node will receive a message 'meant' for it.

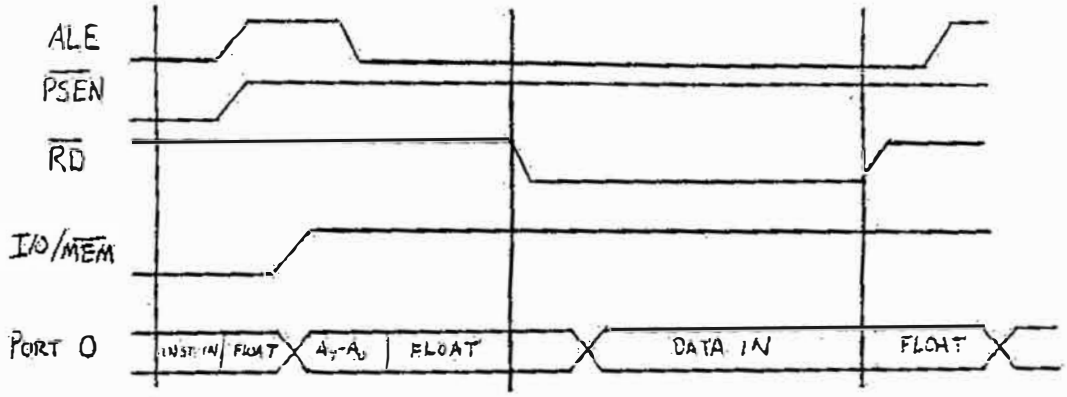
However, since the goal is not that the message follow a predetermined path, but that it follow a closed loop, the messages will simply be repeated over and over until the message gets through following some path. For example, a node that senses the position of a motor (controlled by the network), sends a message repeatedly through the network until it senses that the motor moves, thus closing the loop.

### 3.1.5 Memory Mapped I/O

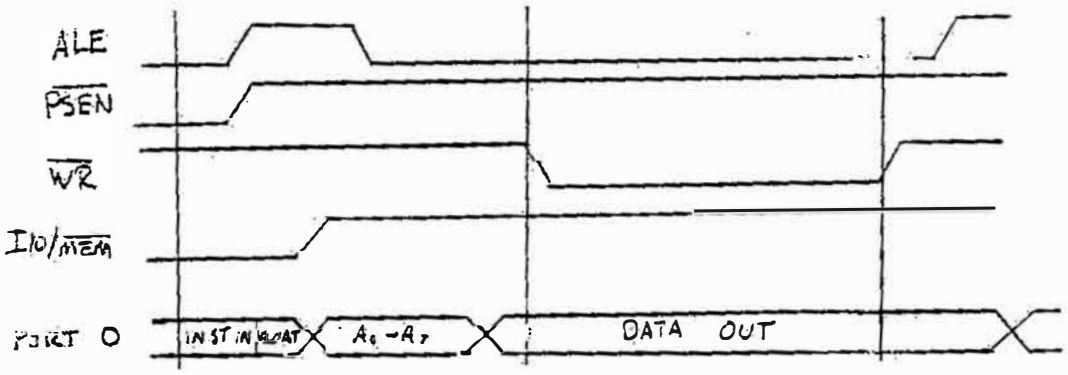
Each node board contains a 34-pin header that allows a ribbon cable to connect I/O devices to the microcomputer's address-data bus, Port 3 pins 2, 3 and 4, and address bit 13 (I/O/MEM'). The Port 3 pins can act as either port pins or as two interrupt inputs (INIT0 and INIT1) and an event counter input (T1). I/O/MEM' is used to enable the I/O devices. This allows a combination of memory mapped, port and interrupt driven I/O.

In memory mapped I/O systems, each I/O device has its own separate address. The device is addressed and gets its data from the processor's external buses as if it were a memory location. Since the device is connected to the microcomputer's address-data bus, it is important that it only places data on that bus when it is told to do so. This means that all I/O devices must include buffers for their data lines. Consequently, when the device is not writing to the bus, it will either be reading the bus or its buffers will be in their high impedance state. The proper time for bus reads and writes is determined by the signals I/O/MEM', ALE, RD' and WR'. The I/O/MEM' line will go high to indicate that an I/O device is being addressed. The particular I/O device addresses (00-FFH) will be valid on the address-data bus on the falling edge of ALE (address latch enable). The device should latch address bits 0-7 on this falling edge. Each I/O device must then compare this address to its own address. If the addresses match, the device can either drive data onto the bus (either by enabling the 3-state buffers or by switching the direction of the bi-directional buffers) on the occurrence of RD'=0 (the processor is doing a read); or read the data from the bus if WR'=0 (the processor has put out valid data). The timing diagrams for I/O read and write are shown in Fig. 3.4.

An I/O device that requires the use of port pins, the interrupt lines or the counter input can be connected directly to the three pins of Port 3. Each of these three pins can be individually programmed as port pins or as their respective special functions (INIT0, INIT1,T1). To use any of the port pins as an output port pin, the data (0 or 1) is




I/O READ CYCLE




I/O WRITE CYCLE

**FIGURE 3.4**

**I/O Timing Diagram**

simply written to the corresponding port bit. If a pin is to be used as an input port pin, a "one" must first be written to the bit; only then the level applied to the pin can be read. To use either Port 3, pin 2 or 3, as interrupt inputs, a "one" must be written to the corresponding bit and the appropriate interrupt enable bit set. To use Port 3 pin 4 as a counter input (T1), a "one" is written to this bit, the counter mode set and the counter started.

Since the 8751 uses Port 3 to generate such signals as WR', RD', TxD, etc., it is important to remember that, whenever Port 3 is used, "ones" must be written to all the other bits in Port 3 so that these pins remain in this special function mode.

For an example of simple I/O devices, see Fig. 3.5. The diagram contains an input device that responds to even addresses (since only address bit 0 is latched) in the range of 6000H to 60FFH. An output device that responds to odd addresses in the same range is also included.

### 3.1.6 Serial I/O

The serial I/O is provided by the 8751's internal serial port which has many operating modes. The mode used by the network to talk to the terminal receives and transmits standard seven bit ASCII and one parity bit (mode one). In order to set this mode, 52H is written to SCON, the

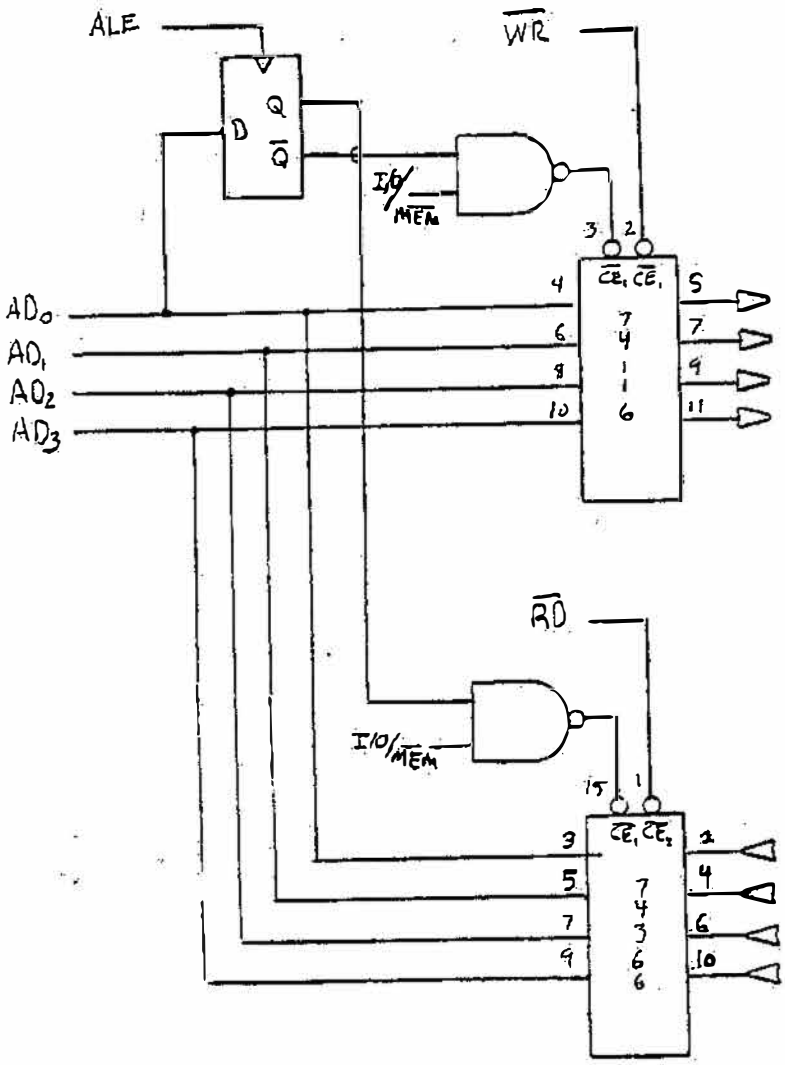



FIGURE 3.5

An Example of an I/O Device

serial port control register. This 8-bit control word establishes mode one, enables the receiver and sets the transmission complete flag (TI). The routine that writes to the serial channel, first tests the TI flag to be sure that the last character has gone out before transmitting the next. It is important that the initialization software sets this flag, so that the first time this routine is called, the transmitter will indicate it is ready.

The baud rate is generated by Timer/Counter 0. This is again a device included in the 8751 chip and is used as an 8-bit counter with auto-reload. The counter begins counting from the reload value held in the register called TH0 and increments on every thirty-second pulse from the T0 input. When the counter overflows, the serial port is clocked, the next bit of the character is transmitted and the counter begins counting at the reload value automatically. The T0 input is fed by the serial clock (SCLK) from the bus with a constant frequency of 153600 hertz (4800 baud x 32). Thus, to get a baud rate of 4800 the reload value (TH0) is set to FFH where it will overflow every count. To get 2400 baud, the serial port should be clocked every other count and therefore the reload value is set to FEH. For the remaining reload values refer to the monitor listing, Appendix A.

To set the auto-reload mode, 60H is written to TMOD (timer mode). Since TMOD holds the mode for both timer/counters, the user must be careful not to change the upper nibble (Timer/Counter 0's mode) when setting the lower nibble if Timer/Counter 1 is being used.

The arbitration of the serial bus is simple. There is a bus line called serial bus busy (SBB), which is driven by each card by open collector NAND gates configured as inverters. These gates are driven by a port bit called serial bus request (SBR). The SBB line is read by a port bit. When a node requests transmission to the terminal, it captures the serial bus by testing the SBB line. If it is low, it must wait and try again later. Otherwise, the SBR bit is set, driving the SBB line low so no one will be able to interrupt. The SBR line is cleared only after the last character is completely transmitted (TI, transmission complete flag, goes high).

### 3.1.7 Power Supply

Since the network is designed to be used both on the lab bench and in a self-contained mobile robot, a variety of power supplies may be used. The simplest way to cope with this variation is to place 5 volt regulators (7805) on each microcomputer node. The only requirement of the bused power is that it remain above 6.5 volts.

## 3.2 The Microcomputer Node Circuit Boards

In order to make the network easily expandable, the cards must be simple to build. Therefore, a printed circuit board was made. This has cut the construction and debug time to one quarter of that required to

build the wirewrapped cards. The wirewrapped prototypes took over 12 hours to build and get running. The construction of the printed circuit board version took approximately 3 hours.

There are two versions of printed circuit boards. The first version (of which only two boards were made), had to be slightly modified by altering traces on the board. Fig. 3.6 shows the chip layout of the final version of the printed circuit board. Fig. 3.7 has the BCL schematic with pin numbers for this printed circuit board layout.

For a detailed description of the construction of a microcomputer node circuit board, see Appendix C.

### 3.2.1 Prototype Boards

There are three wirewrapped boards that were used as prototypes. These boards are functionally the same although the chip layout and wiring of individual gates is different. Fig. 3.8 shows the chip layout for the prototype boards and Fig. 3.9 has the BCL schematic with pin numbers for the prototype boards. The I/O bus connection is also different. The prototypes use a 50-pin header instead of the 34-pin header used on the printed circuit cards. If these three boards are to be used in the future, it may be wise to change these connectors to make them compatible.

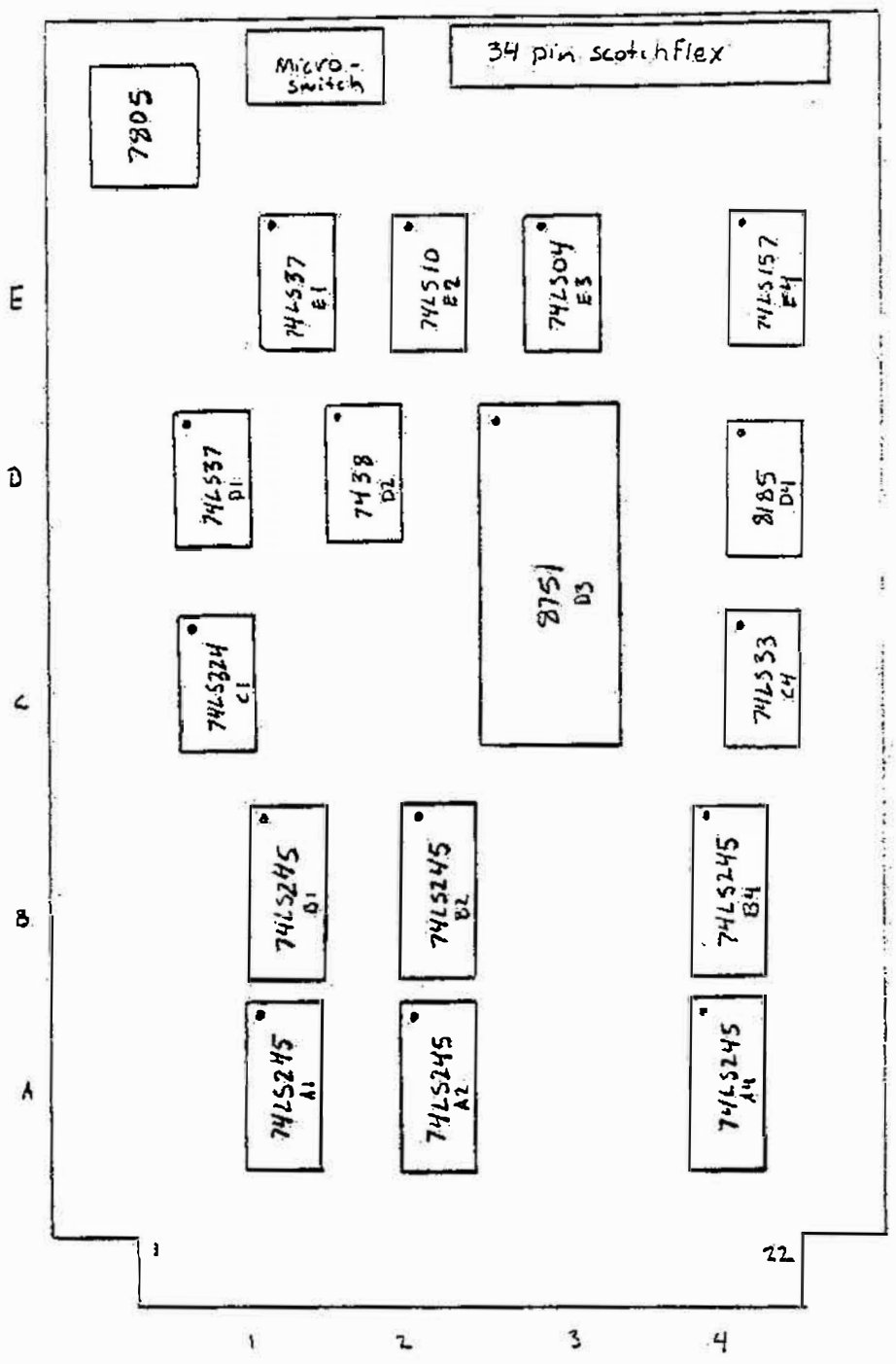



FIGURE 3.6

Printed Circuit Board Chip Layout

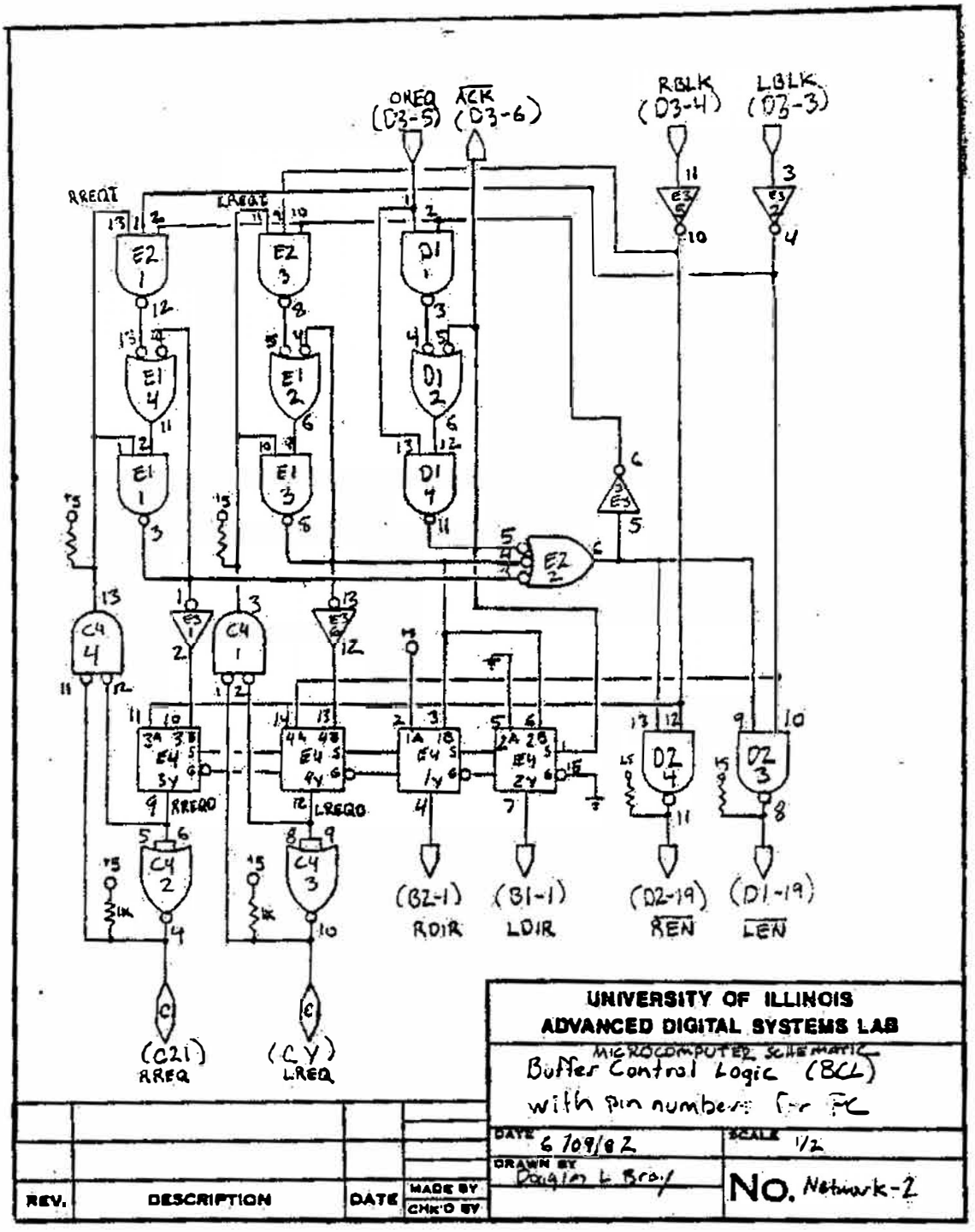



FIGURE 3.7

BCL Schematic with Pin Numbers for Printed Circuit Board

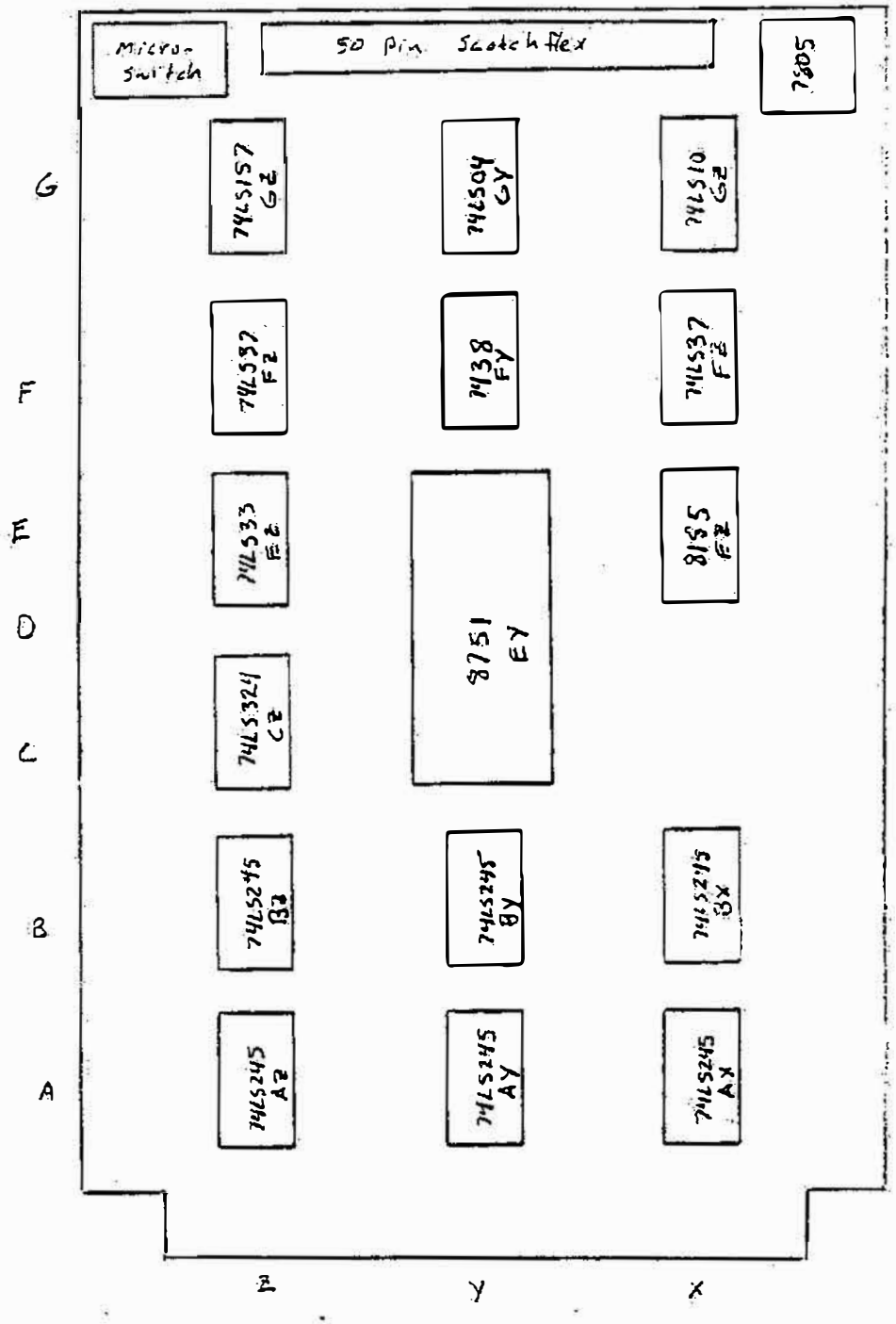



FIGURE 3.8

Chip Layout for Prototype Board

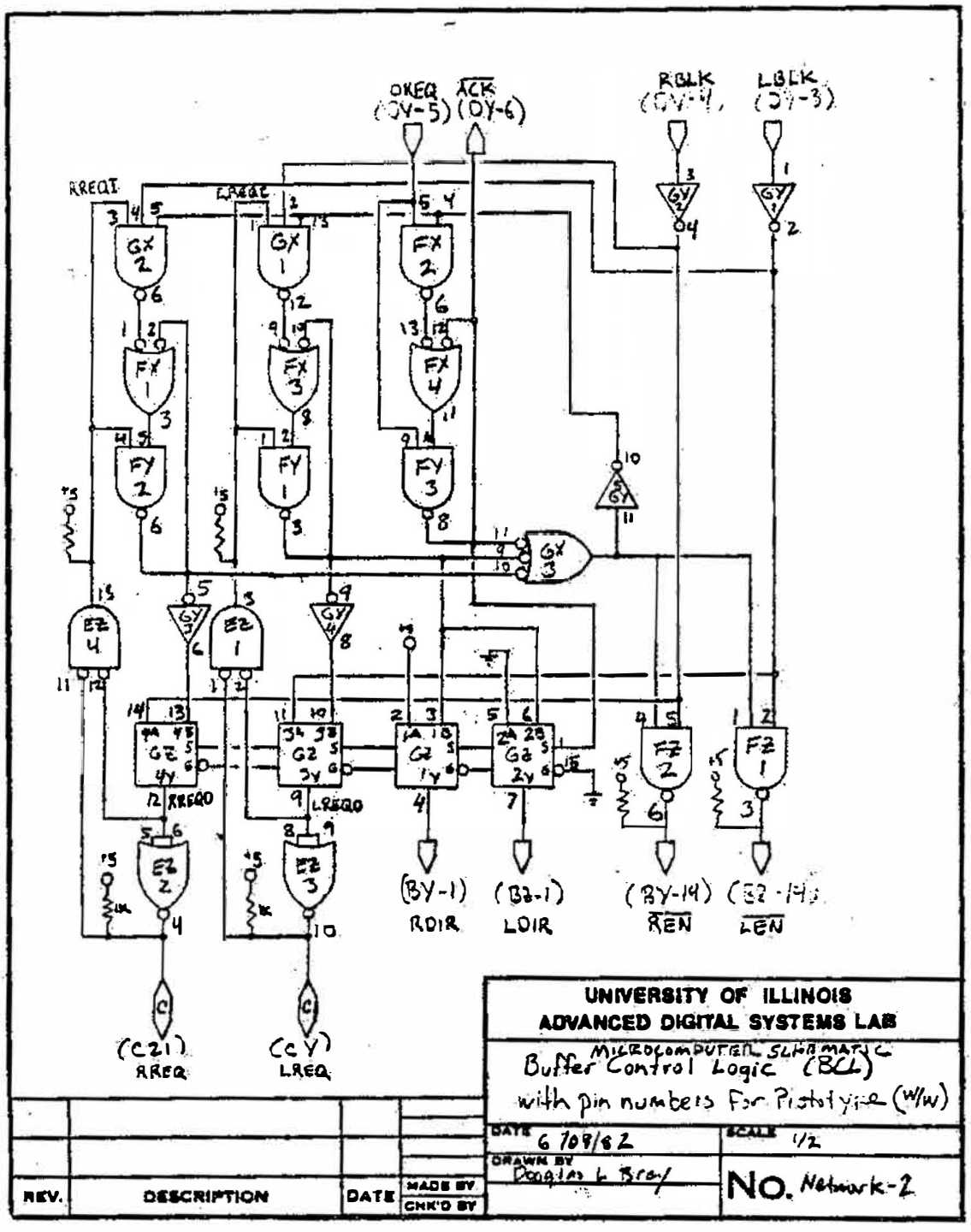



FIGURE 3.9

BCL Schematic with Pin Numbers for the Prototype

## 3.3 Bus Topology and Backplane

The bus topology, as mentioned in the system overview, is three dimensional. The topology is a two by two by two cube, i.e., a cube made of eight smaller cubes. Since the microcomputer nodes are placed in the links between vertices of the cubes, there are three planes of 12 nodes with two sets of nine nodes interconnecting the planes for a total of 54 nodes. This topology can be seen in Fig. 3.10. The way in which the 3-D bus is wired into a 2-D backplane is shown in Fig. 3.11.

It is important to note that only the PBUS and the neighbor request lines must be wired in this special fashion; all other lines may be wired in a standard linear manner. Fig. 3.12 shows the bus pin definition. The pins labeled with an 'L' are those that can be wired in a linear bus. At present the bus is wired for 12 cards.

## 3.4 Terminal Interface and Master Reset Board

The terminal interface and master reset board is a small wirewrapped board mounted inside the network rack structure, but it does not take up a bus slot. This board contains the interface between the RS-232 terminal and the TTL compatible RxD and TxD lines of the 8751. The clock used to generate the baud rate and the common reset are also provided by this board's circuitry.

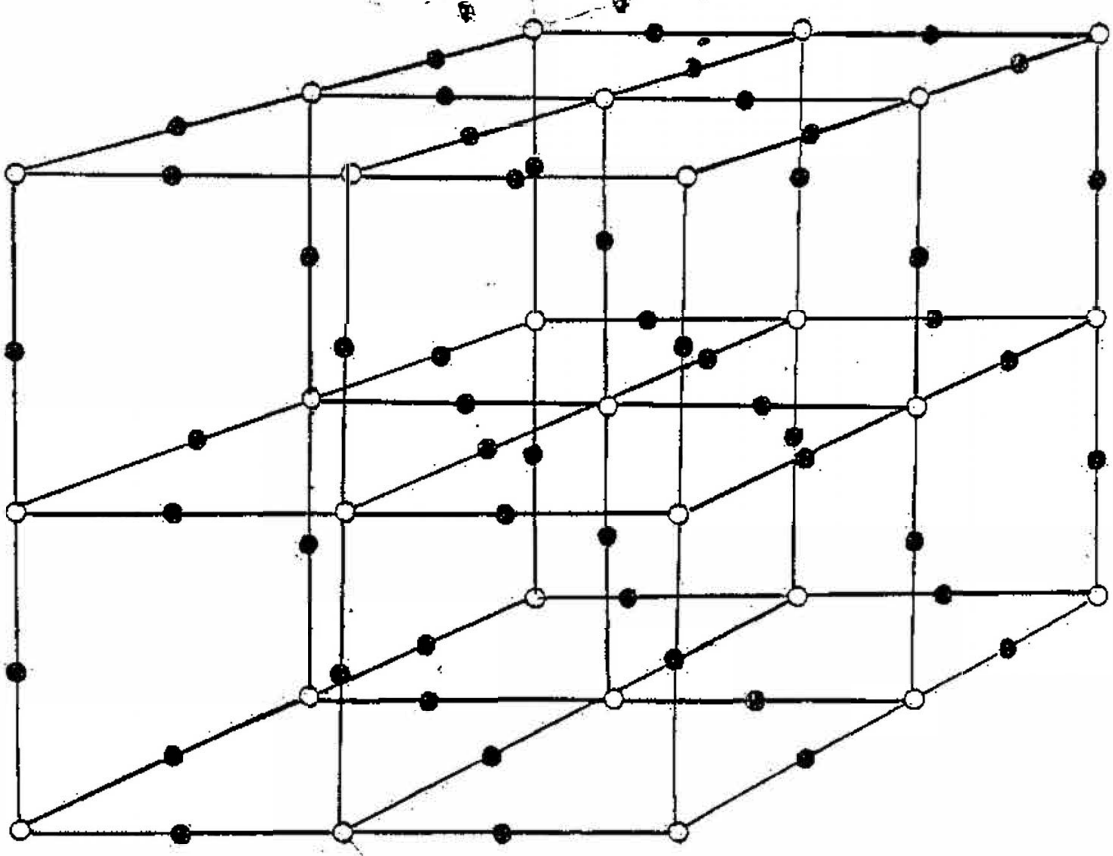


FIGURE 3.10

3-Dimensional Bus Structure

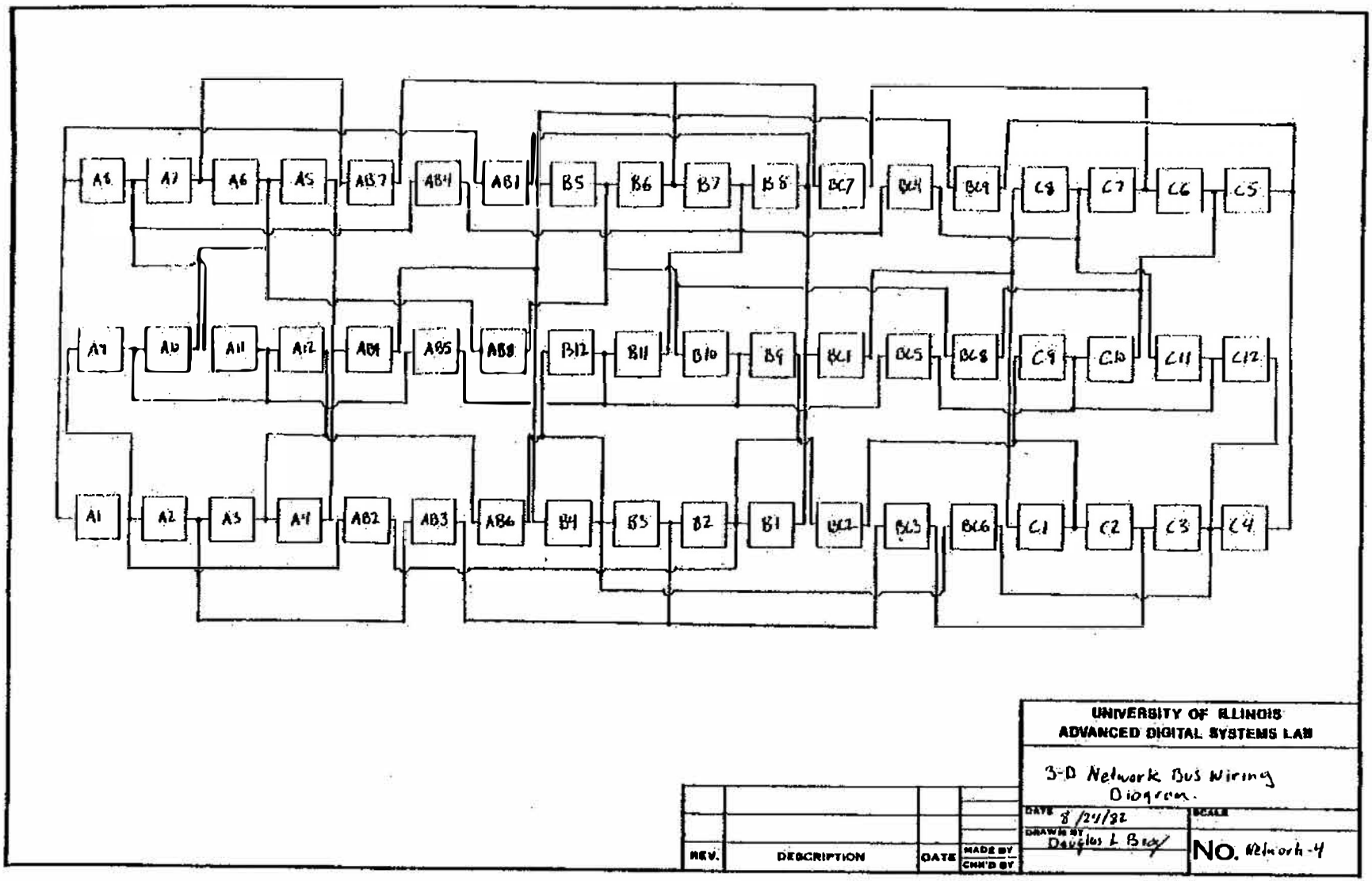



FIGURE 3.11

Bus Wiring Diagram

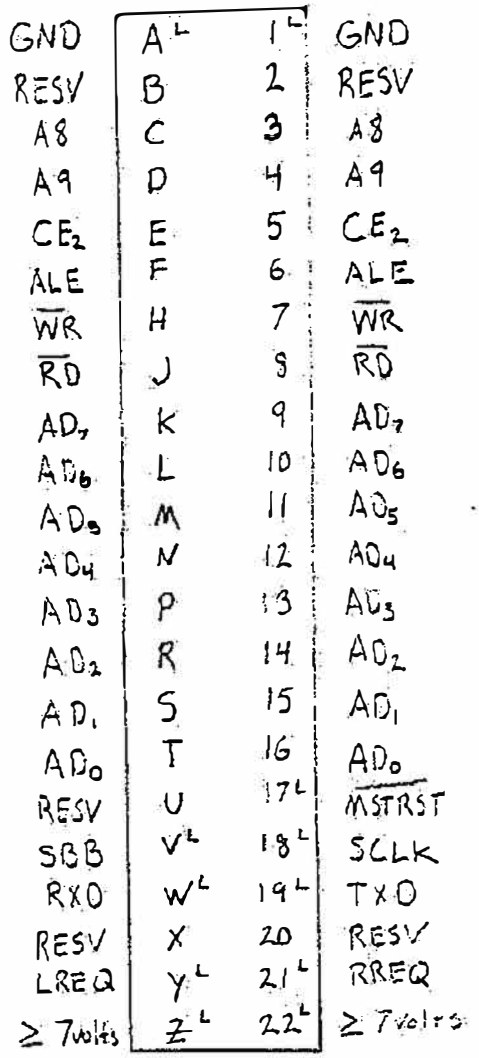



**FIGURE 3.12**

**Bus Pin Definition**

The RS-232 interface is realized with a National Semiconductor 1488 line driver which takes TTL signals and converts them to RS-232 signals required by the terminal, and by a National Semiconductor 1489 line receiver which takes the RS-232 signals from the terminal and converts them to TTL logic levels.

The serial clock (SCLK) generator feeds the signal from a 2 megahertz crystal osillator into a 74LS161 counter which divides the frequency by 13. The resulting 153600 hertz clock is the SCLK line on the bus.

The master reset, including power up reset, and a front panel reset switch, is realized with a 74LS123 monostable multivibrator. Inputs to trigger the 100 microsecond pulse are a debounced front panel push button and an RC power up circuit. Fig. 3.13 includes the schematic diagram for this board.

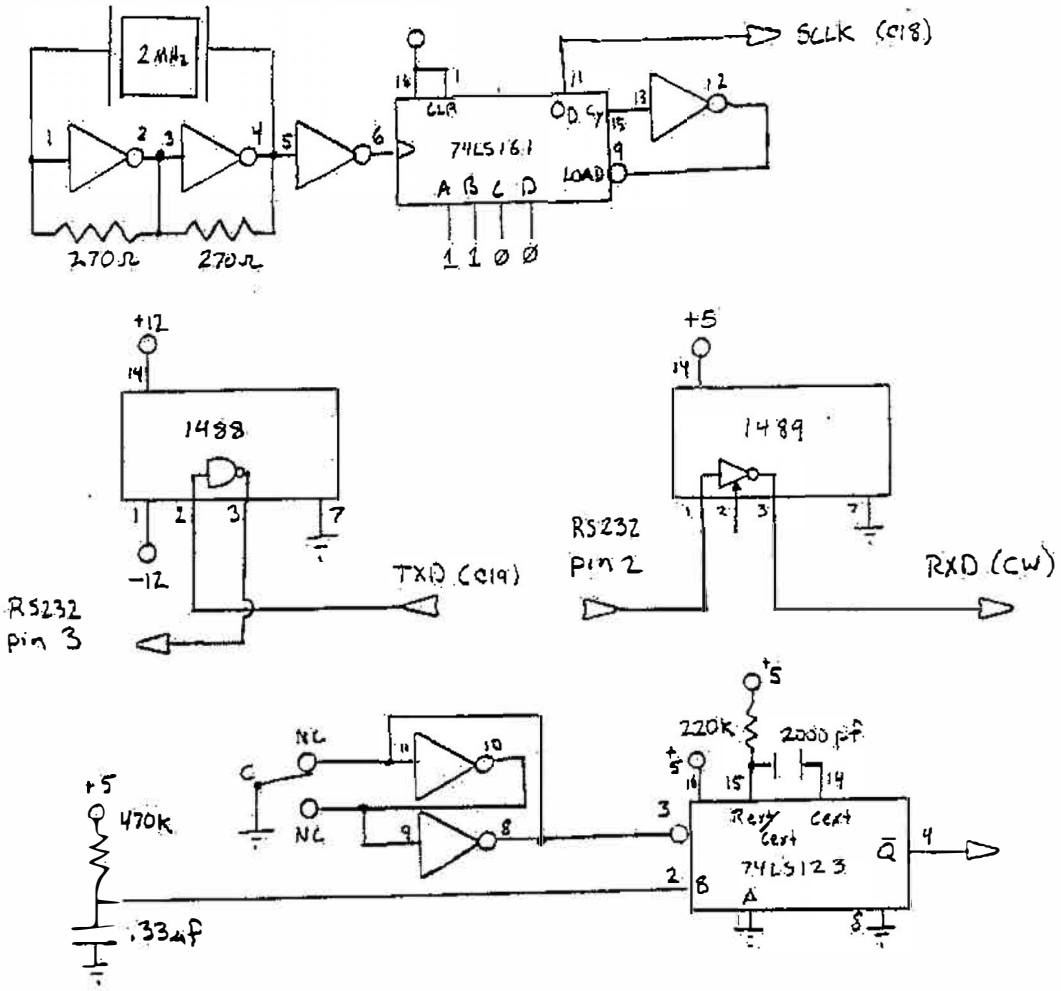



FIGURE 3.13

Termnial Board Schematic

# CHAPTER 4

## SYSTEM SOFTWARE

All software in the network is resident in the 4048 bytes of erasable programmable read only memory (EPROM) internal to each 8751. This software is in two sections: the system software or the monitor and the user programs. Whenever the 8751 is programmed with a new user program, the monitor must be linked with that program and burned along with it.

The monitor has commands that will aid the user in debugging and executing his programs. This software also contains many utility routines that may be called by the user for such functions as inter-node interaction and terminal interactions.

## 4.1 Invoking the Monitor

To invoke the monitor of a particular microcomputer node, after reset, type CONTROL N followed by the two hex-digits (one-byte) of the name of that node. For example, to enter the monitor of the node named 1C. Type CONTROL N, 1, and C. The node will prompt the user with 1C:. At this point, the user enters one of the commands described below.

## 4.2 The list of Monitor Commands

BLOCK - Bn (n=L or R) - This command blocks the transactions coming in from the left (n=L) or the right (n=R) of the current node by closing the corresponding buffers. Entering the two commands BL and BR will totally isolate the node.

UNBLOCK - Un (n=L or R) - This command is the inverse of BLOCK. It opens the buffers closed by BLOCK.

CHANGE BAUD RATE - Cn (n=0-6) - This command changes the node's baud rate to a value corresponding to n. On reset, the baud rate is 4800.

| n | baud |
|---|---|
| 0 | 4800 |
| 1 | 2400 |
| 2 | 1200 |
| 3 | 600 |
| 4 | 300 |
| 5 | 150 |
| 6 | 110 |

GO - Gn (n= code address) - This command excecutes the program at address n.

RUN USER PROGRAM - R - This command executes the user program given the label USER at assembly time. The starting address of the user program must be given the label USER, and USER must be declared PUBLIC for the linker. This will be explained further in section 4.5.

DISPLAY CODE - D 'from' start-address 'to' end-address - This command displays the contents of the code memory from start-address to end-address. CONTROL C will exit the display and any other key will stop and start the display for viewing. The words 'from' and 'to' are printed by the monitor.

SUBSTITUTE DATA MEMORY - SDn (n= data address (00-7FH)) - SDn allows the user to read and change, if desired, the contents of any data address. Since the accumulator (ACC) and all other registers of the 8751 are simply data addresses, this command will allow the user to change the contents of these registers. One should take care, however, since one may change values that will crash the monitor. The command will print the data address, an equal sign (=) and the data value. You can then enter i) a new value, ii) a space to look at the next location, or iii) a carriage return to exit the command.

SUBSTITUTE EXTERNAL MEMORY - SXn (n= external address) - This command is identical to SDn with the exception that mailboxes or I/O are

read from or written to.

n= 1000H-13FFH - read this node's mailbox and write to all other mailboxes.

5000H-53FFH - read the mailbox and write to ONLY this node's mailbox.

6000H-60FFH - read and write to I/O

EXIT - E - This command exits the present node and allows the CONTROL N procedure to enter another node's monitor.

4.3 List of Utiltiy Routines

WRMES - WRITE MESSAGE - This routine writes the contents of the ACC to the mailbox or I/O device with the address held in the DPTR.

RDMES - READ MESSAGE - This routine reads the contents of the address in the DPTR ORed with 4000H (to make it "private") and places it in the ACC. However, the DPTR is not changed.

GETSB - GET THE SERIAL BUS - This routine waits for the SBUS to be idle and captures it.

RELSB - RELEASE THE SERIAL BUS - This routine waits for the last character to be transmitted and relinquishes the serial bus.

SPOUT - SERIAL PORT OUT - This routine outputs the character's ASCII code held in the ACC to the serial port.

SPIN - SERIAL PORT IN - This routine inputs a character from the serial port and places the ASCII value in the ACC.

ECHO - ECHO ASCII VALUE - This routine waits for a character to come in, places the ASCII value into the ACC, and echos it to the terminal.

HEXOUT - HEX VALUE OUT - This routine takes the hex value from the ACC, converts each nibble to ASCII and outputs them to the serial port.

CROUT - CRT OUT - This routine gets the serial bus, outputs to the terminal the node's name, a dash (-) and the hex value in the ACC converted to ASCII. This is useful in printing values for program debugging.

HEXIN - HEX VALUE IN - This routine takes 0 to 4 characters in from the terminal followed by a carriage return or space and converts them to hex. The upper byte is left in the multiplication register (B) and the lower byte is left in the ACC. If less than 4 characters are present, leading zeros are filled in. It also captures and releases the serial bus as needed.

SPMES - OUT SERIAL PORT MESSAGE - This routine outputs a character string addressed by the DPTR. The end character of the string must be

00. The assembler allows definition of strings, including carriage return and line feed as follows:

```
MESSAGE: DB 13,10,'Example Message',13,10,00
```

To call SPMES:

```
MOV DPTR,#MESSAGE
CALL SPMES.
```

NEWLIN - NEW LINE - This routine outputs a carriage return and a line feed. It is assumed that the node already has control of the SBUS.

NAM? - IS NAME CALLED? - If a CONTROL N, and the node's name are received, this routine exits the user program and jumps to the monitor. Otherwise, the routine does a return (RET) to the user program.

## 4.4 Requirements of User Programs

In order to link a user program with the monitor the user must:

1. Name the node by equating the label NAM to the desired one-byte name.

2. Label the entry point (starting address) of the user program with USER.

3. Declare USER and NAM as PUBLIC labels as follows:

```
PUBLIC USER,NAM
```

4. Declare any monitor routines or variables used, as EXTERNAL as follows:

```
EXTRN CODE (WRMES,RDMES,ECHO,etc.)
EXTRN DATA (INMES,etc.)
```

5. Release the serial bus. Since the serial bus has been captured before entering the user program, the user program must release the bus when done with it, e.g., after terminal input.

For detailed instructions on how to assemble and link user programs, refer to Appendix D.

# CHAPTER 5

## A DEMONSTRATION

After proving the network functional using small test programs, it was decided that a demonstration of how the network might be used by future students was needed. Although this demonstration uses only a few nodes and facilitates only a few features of the network, it does show the potential that this network has for investigating the interaction between this type of entity and its environment.

It was decided that an appropriate means to allow this entity, the network, to interact with its environment and other entities, people in this case, was with light and a small motor. A light sensing and seeking creature (Seeker) was created. Seeker, like humans, has both a fuzzy peripheral and a focused narrow field of vision. However, Seeker has these two perceptive properties in two separate systems.

The first system is a ring of unfocusable eyes that give it 360-degree peripheral vision. For focused perception, Seeker has an eye on an arm or hand that can be aimed at any point in its 360-degree field of vision. Seeker's perceptive apparatus lacks the ability to 'feel' where its hand is. In order for it to know where its hand is, Seeker must be able to see its hand with the peripheral vision ring. Due to the poor discrimination of detail that the ring has, Seeker must wave its hand to be sure that it is, in fact, seeing its hand. Seeker has the ability to learn or be trained to avoid looking at a certain light intensity when that 'bad' intensity is presented coincident with pain.

To make Seeker operational, the network with four nodes and two previously built hardware devices were used. The first device was a light detection system, consisting of a ring of 32 photo sensitive transistors and the corresponding analog to digital conversion hardware, designed and constructed by Kevin W. Warren. The second piece of hardware was a stepping motor driver circuit designed by Martin Eberhard. The stepping motor, with the 'hand', was mounted in the center of the light detection ring. The hand holds another photo-transistor and a small lamp aimed back toward the photo-sensor ring. This allows the hand to be 'seen' by the creature itself. Fig. 5.1 displays this setup.

The observer (a human entity) interacts with the creature by shining a flashlight on one of the photo-sensors of the ring. Seeker discriminates between the flashlight and its own hand by realizing that

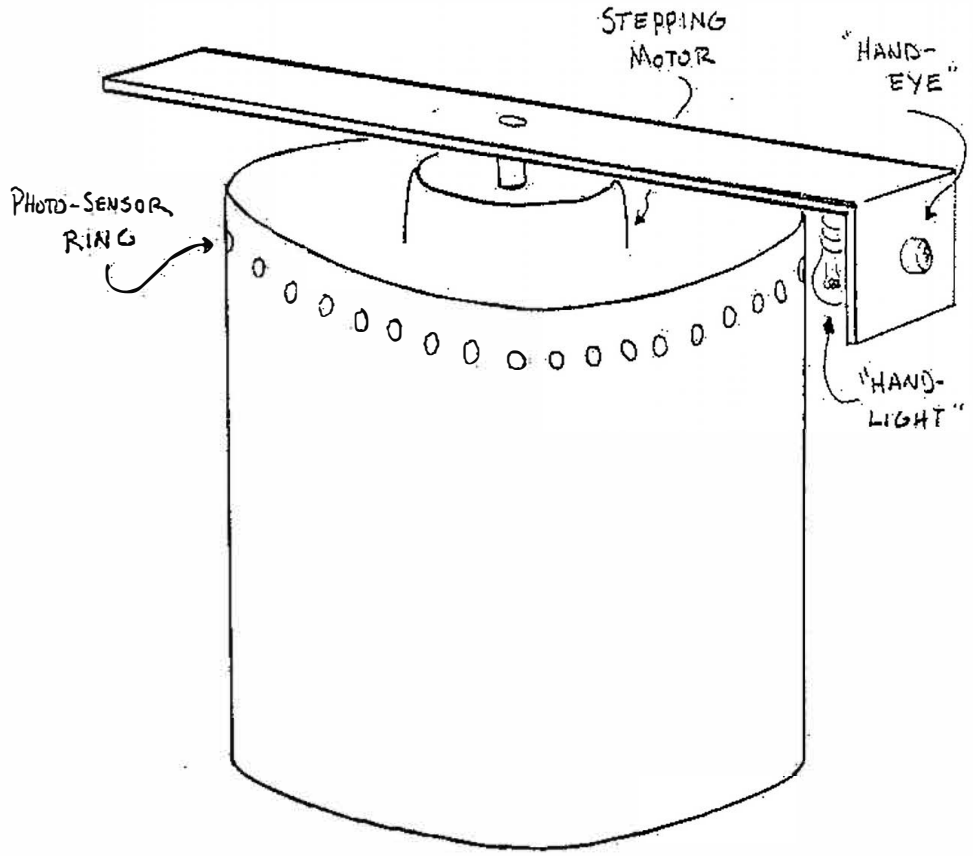



FIGURE 5.1

Demonstration Hardware Setup

moving the hand will cause one photo-sensor of the ring to turn off and another to turn on. Therefore, the hand must be pointing at the one that came on. Now that the creature knows where the hand and the flashlight are, Seeker moves its hand toward the flashlight at a rate proportional to the distance between the hand and the light. As in the example of a person reaching for an object, the loop is closed and a new distance is 'seen', the error is reduced and the speed of the hand is slowed down until the hand stops pointing at the flashlight. Seeker can now be taught that the current intensity of the flashlight is the 'bad' value by depressing the 'pain' button. The 'reflex' occurs whereby the hand is turned 90 degrees away from the light and the new 'bad' value is remembered. Otherwise, if the value read by the hand photo-sensor is the 'bad' value previously taught, the network will execute the 'reflex' action.

Four network nodes were used to perform this demonstration: one to sample all the photo-sensors, one to control the motor and two to implement the learning. The interaction between the nodes and the corresponding closed loops is shown in Fig. 5.2. The photo-sensor sampling node is responsible for sending a message, representing a speed and a direction, to the motor controlling node. It also must send a message that represents the light intensity detected by the hand 'eye' to Learning Node One. Learning Node One one is responsible for sending the message to the motor node to indicate that the reflex action is to be taken. It also relays the message that represents the 'bad' intensity to Learning Node Two. This represents the first half of the

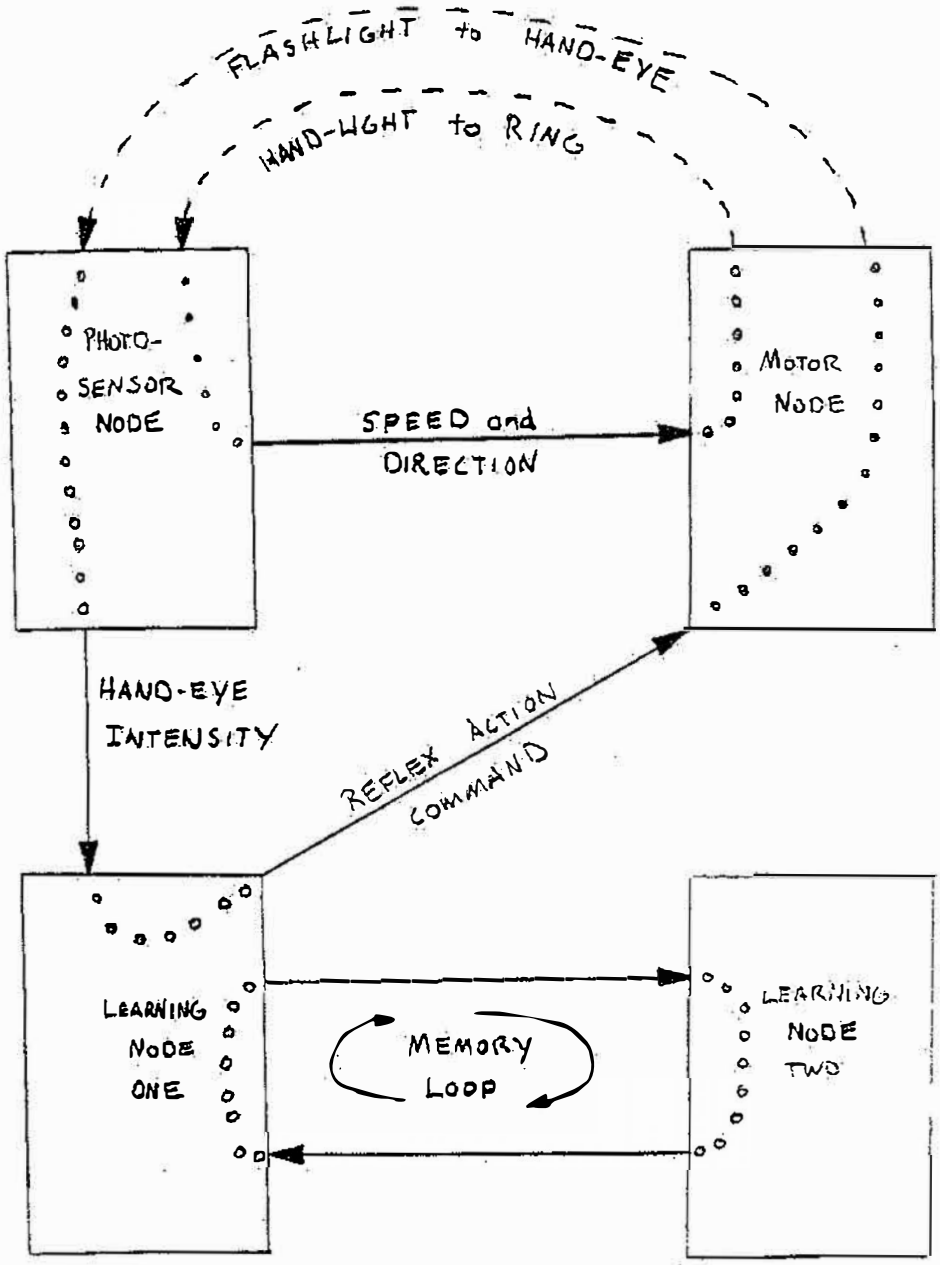



FIGURE 5.2

Node Interaction for the Demonstration

loop which constitutes the memory of the 'bad' intensity value. The loop is closed when Learning Node Two reads the value and sends it back to be read by Learning Node One which, of course, continues the loop by returning the value to Learning Node Two, unless a new 'bad' value is taught. The dashed lines of Fig. 5.2 show how the loops, set up by the inter-node interaction, are completed through the environment.

The loop marked 'hand-light to ring' corresponds to feedback that occurs when the ring 'sees' the hand move, which, in turn, changes the speed of the hand's movement. While the hand is moving toward the flashlight, a number representing the distance between the hand and the flashlight is sent out into the network. Notice that the meaning of this message changes from a distance, as interpreted by the photo-sensor node, to a speed when read by the motor node. The motor turns the hand at that speed making the distance smaller which, in turn, lessens the speed. The hand will slow as it approaches its target and will stop when the hand-eye 'sees' the flashlight.

The other loop that includes the environment is slightly different. It is not a continous loop as above, but rather it is closed only when the hand is pointing at the flashlight. It is here that the photo-sensor node transmits the intensity value of the light to Learning Node One. If this value compares with the value being remembered by the 'inside' loop, the 'reflex' message is sent out to be read by the motor node.

To reiterate, the 'inside' loop remembers the 'bad' value. When Learning Node One senses that a switch, which is connected to one of three I/O port pins, is pressed, it passes the value from the hand-eye to Learning Node Two which closes the loop by sending it back to Learning Node One and the loop is continued. This loop can be thought of as a 'vision' or a remembrance, as in humans when recalling past events. Notice, that the activation or reactivation of this inside loop recreates the sensory experience, even if the loop is activated artificially by feeding a number into the loop. The creature will react to the corresponding light intensity as if it had been trained with a real flashlight.

These inside loops, consisting of nodes with no direct connection to the environment, will be very important and prevalent in the future of this network. They will be responsible for a large amount of the decision making process.

# CHAPTER 6

## CONCLUDING REMARKS

Since the completion of the demonstration, this network has been ready for use and expansion by students of the Advanced Digital Systems Laboratory in a variety of educational uses. There are now, including the three wirewrapped prototype boards, seven tested and working nodes. This should be enough to provide a starting point for several students to put the network to use in different projects.

Plans have been made to interface these nodes with a variety of environmental interaction devices, such as ultra-sonic range sensors and motor-driven wheels. These will allow the network to be part of a self-contained mobile robot. There are also several other projects within the lab that could be interfaced to the network. A robotic arm, voice synthesis system, artificial vision system, and others would increase the network's ability to interact with its environment and with us.

Furthermore, it is the author conviction that much is to be gained from approaching microcomputer networks as presented here; that is, by placing the emphasis on the network as the new entity, thereby allowing its full potential to develop. We must go beyond the insistence in a network made of "efficient" microcomputers.

It is also the author conviction that the software development needed to perform a specific task is simplified when more and more small pieces of program run in concurrent microcomputers (as this network), as opposed to fewer larger programs in a conventional network. We look forward to the strengthening of these statements through the work that will be performed by future students.

# APPENDIX A

# MONITOR LISTING

```
LOC  OBJ              LINE     SOURCE

                        1     $TITLE(ADSL 8751 NETWORK MONITOR  21-JULY-82)
                        2
                        3     ;TO INVOKE THE MONITOR:
                        4     ;FIRST TYPE CONTROL N FOLLOWED THE NUMBER OF THE NODE YOU WISH TO
                        5     ;CONTROL. THAT NODE WILL PROMPT YOU WITH ITS NUMBER AND ':'.
                        6     ;ENTER A COMMAND:
                        7     ;       BN- WILL BLOCK EITHER THE LEFT (N=L) OR RIGHT (N=R)
                        8     ;       UN- WILL UNBLOCK AS ABOVE
                        9     ;       CN- CHANGE THE BAUD RATE (N=0-6) SEE SOFTWARE FOR RATES
                       10     ;       D FROM X TO Y- DISPLAY CODE MEMORY
                       11     ;       GN- EXECUTE PROGRAM AT N
                       12     ;       R- EXECUTE USER PROGRAM
                       13     ;       SDN- SUBSTITUTE DATA MEMORY
                       14     ;       SXN- SUBSTITUTE EXTERNAL MEMORY
                       15     ;       E - WILL RETURN YOU TO THE CONTROL N POINT TO TALK TO ANOTHER NODE
                       16
                       17     EXTRN CODE (USER)
                       18     EXTRN NUMBER (NAM)
                       19     PUBLIC  CMD
                       20
  0097                 21     SBR     BIT     P1.7
  0096                 22     SBB     BIT     P1.6
                       23
----                   24             CSEG    AT 0
0000 020010            25             JMP     START
                       26
----                   27             CSEG    AT 10H
0010 758140            28     START:  MOV     SP,#40H         ;SET THE STACK POINTER
0013 120189            29             CALL    INIT            ;INITIALIZE PORTS AND VARIABLES
0016 1201E0            30             CALL    SPINIT          ;INITIALIZE SERIAL PORT
0019 7400      F       31             MOV     A,#NAM
001B 54F0              32             ANL     A,#0F0H         ;GET UPPER BYTE OF NAME
001D C4                33             SWAP    A
001E 90028A            34             MOV     DPTR,#HEXTBL
0021 93                35             MOVC    A,@A+DPTR       ;CONVERT TO ASCII
0022 F509              36             MOV     NAMH,A          ;SAVE UPPER BYTE OF NAME
0024 7400      F       37             MOV     A,#NAM
0026 540F              38             ANL     A,#0FH          ;GET LOWER BYTE OF NAME
0028 93                39             MOVC    A,@A+DPTR       ;CONVERT TO ASCII
0029 F50A              40             MOV     NAML,A          ;SAVE LOWER BYTE OF NAME
                       41
002B 12023E            42     NWAIT:  CALL    RELSB           ;BESURE SERIAL BUS IS RELEASED
002E 12024B            43             CALL    ISNAM           ;IF CONT. N AND NAME JMP TO CMD
0031 2090F7            44             JB      P1.0,NWAIT      ;ELSE WAIT FOR IT, UNLESS
0034 020000    F       45             JMP     USER            ;YOU WANT TO GO DIRECTLY TO USER
                       46                                     ;(8751 PIN 1 IS TIED LOW)
0037 12022D            47     CMD:    CALL    GETSB           ;GET THE SERIAL BUS
003A 120203            48             CALL    NEWLIN
003D 7400      F       49             MOV     A,#NAM          ;OUTPUT PROMPT
003F 12026B            50             CALL    HEXOUT
```

```
0042 743A          51          MOV     A,#':'
0044 1201F1        52          CALL    SPOUT
0047 1201EC        53          CALL    ECHO            ;INPUT COMMAND
004A 90007B        54          MOV     DPTR,#CMDTBL-3  ;SETUP TO POINT TO TABLE
004D F508          55          MOV     TEMP,A          ;SAVE THE COMMAND
004F E4            56          CLR     A
0050 C0E0          57          PUSH    ACC
0052 D0E0          58  CMDL:   POP     ACC
0054 04            59          INC     A
0055 04            60          INC     A
0056 04            61          INC     A               ;POINT INDEX TO NEXT TABLE ENTRY
0057 B41B03        62          CJNE    A,#TBLEND-CMDTBL+3,CONT
005A 020074        63          JMP     ERROR           ;IF CMD NOT FOUND
005D C0E0          64  CONT:   PUSH    ACC
005F 93            65          MOVC    A,@A+DPTR       ;ELSE COMPARE NEXT CMD
0060 8508EF        66          CJNE    A,TEMP,CMDL
                   67
0063 D0E0          68          POP     ACC
0065 04            69          INC     A               ;ONCE THE COMMAND IS FOUND
0066 C0E0          70          PUSH    ACC
0068 93            71          MOVC    A,@A+DPTR       ;GET JMP ADDR
0069 FB            72          MOV     R3,A
006A D0E0          73          POP     ACC
006C 04            74          INC     A               ;POINT TO HIGH(ADDR)
006D 93            75          MOVC    A,@A+DPTR
006E F582          76          MOV     DPL,A
0070 8B83          77          MOV     DPH,R3          ;NOW POINT TO ROUTINE ADDR.
0072 E4            78          CLR     A
0073 73            79          JMP     @A+DPTR
                   80
0074 743F          81  ERROR:  MOV     A,#'?'          ;IF ILLEGAL CMD OUTPUT ?
0076 1201F1        82          CALL    SPOUT
0079 120203        83          CALL    NEWLIN
007C 80B9          84          JMP     CMD             ;AND GET NEW INPUT
                   85
007E 42            86  CMDTBL: DB      'B'
007F 0096          87          DW      BLKRTN  ;BLOCK CMD
0081 43            88          DB      'C'
0082 01A0          89          DW      BAUD    ;CHANGE BAUD RATE CMD
0084 44            90          DB      'D'
0085 0130          91          DW      DISCOD  ;DISPLAY CODE MEMORY CMD
0087 45            92          DB      'E'
0088 002B          93          DW      NWAIT   ;EXIT CMD
008A 47            94          DB      'G'
008B 0193          95          DW      GO      ;EXEC PROG CMD
008D 52            96          DB      'R'
008E 0000      F   97          DW      USER    ;RUN USER PROGRAM
0090 53            98          DB      'S'
0091 00BC          99          DW      SUBMEM  ;SUB. DATA MEMORY CMD
0093 55           100          DB      'U'
0094 00A9         101          DW      UNBLK   ;UNBLOCK CMD
                  102
                  103  TBLEND:                         ;FOR CALCULATING TABLE LENGTH
0096 1201EC       104  BLKRTN: CALL    ECHO            ;INPUT R OR L
0099 120203       105          CALL    NEWLIN
```

```
009C B45205          106                  CJNE    A,#'R',LEFTB     ;IF NOT R MUST BE L
009F 430B08          107     RIGHTB:      ORL     BLOCK,#08        ;BLOCK THE RIGHT
00A2 8093            108                  JMP     CMD
00A4 430B04          109     LEFTB:       ORL     BLOCK,#04        ;BLOCK THE LEFT
00A7 808E            110                  JMP     CMD
                     111
00A9 1201EC          112     UNBLK:       CALL    ECHO             ;JUST LIKE ABOVE
00AC 120203          113                  CALL    NEWLIN
00AF B45205          114                  CJNE    A,#'R',LEFTU
00B2 530B04          115     RIGHTU:      ANL     BLOCK,#04
00B5 8080            116                  JMP     CMD
00B7 530B08          117     LEFTU:       ANL     BLOCK,#08
00BA 0137            118                  JMP     CMD
                     119
                     120
00BC 1201F9          121     SUBMEM:      CALL    SPIN             ;INPUT 'X' OR 'D'
00BF B44403          122                  CJNE    A,#'D',SX?
00C2 0200C8          123                  JMP     SMOK
00C5 B458F4          124     SX?:         CJNE    A,#'X',SUBMEM    ;IF NEIGTHER TRY AGAIN
00C8 1201F1          125     SMOK:        CALL    SPOUT            ;ECHO THE 'X' OR 'D'
00CB FA              126                  MOV     R2,A             ;SAVE 'X' OR 'D'
00CC 12029A          127                  CALL    HEXIN            ;BRING IN ADDR TO SUB
00CF 120203          128                  CALL    NEWLIN
00D2 F582            129                  MOV     DPL,A            ;SET UP STARTING ADDR.
00D4 85F083          130                  MOV     DPH,B
00D7 F8              131                  MOV     R0,A             ;PUT ADDR IN R0
00D8 780A            132     SBM0:        MOV     R3,#0AH          ;SET COUNTER FOR NEWLINE
00DA BA5805          133     SBM1:        CJNE    R2,#'X',SBM2
00DD E583            134                  MOV     A,DPH
00DF 12026B          135                  CALL    HEXOUT           ;OUTPUT UPPER BYTE OF ADDR IF 'X'
00E2 E8              136     SBM2:        MOV     A,R0             ;A <= NEW ADDR
00E3 B45802          137                  CJNE    A,#'X',SBM25
00E6 E582            138                  MOV     A,DPL
00E8 12026B          139     SBM25:       CALL    HEXOUT
00EB 743D            140                  MOV     A,#'='
00ED 1201F1          141                  CALL    SPOUT
00F0 E6              142                  MOV     A,@R0            ;GET THE DATA
00F1 BA5803          143                  CJNE    R2,#'X',SBM3
00F4 1201D5          144                  CALL    RDMES            ;IF 'X' GET DATA FROM XRAM
00F7 12026B          145     SBM3:        CALL    HEXOUT
00FA 7420            146                  MOV     A,#' '
00FC 1201F1          147                  CALL    SPOUT
00FF 3098FD          148                  JNB     RI,$
0102 E599            149                  MOV     A,SBUF
0104 C2E7            150                  CLR     ACC.7            ;INPUT A CHAR BUT DONT CLR RI
                     151                                           ;AFTER OUTING ADDR=DATA
                     152                                           ;INPUT CR,SP OR NEW DATA
0106 B42003          153                  CJNE    A,#' ',SBM4      ;IF SP GOTO NEXT ADDR
0109 020125          154                  JMP     SMNEXT
010C B40D04          155     SBM4:        CJNE    A,#0DH,SBM5      ;IF CR EXIT
010F C293            156                  CLR     RI               ;NOW CLR RI
0111 0137            157                  JMP     CMD
0113 12029A          158     SBM5:        CALL    HEXIN    ;SINCE RI WAS NOT CLR'D IT WILL USE THE INPUT AS
                     159                                   ;ITS 1ST CHAR RATHER THAN WAITING FOR A NEW ONE
0116 BA5806          160                  CJNE    R2,#'X',SBM6
```

```
0119 1201C3          161              CALL    WRMES           ;WRITE DATA TO EITHER XRAM OR
011C 020120          162              JMP     SBM7
011F F6              163      SBM6:   MOV     @R0,A           ;TO DATA RAM
0120 7420            164      SBM7:   MOV     A,#' '
0122 1201F1          165              CALL    SPOUT
0125 C298            166      SMNEXT: CLR     RI              ;CLR RI SINCE WE NEED ANOTHER INPUT
0127 A3              167              INC     DPTR
0128 08              168              INC     R0              ;GET NEXT ADDR
0129 DBAF            169              DJNZ    R3,SBM1
012B 120203          170              CALL    NEWLIN
012E 80A8            171              JMP     SBM0
                     172
0130 900187          173      DISCOD: MOV     DPTR,#STAMES
0133 12031D          174              CALL    SPMES
0136 12029A          175              CALL    HEXIN
0139 F8              176              MOV     R0,A
013A A9F0            177              MOV     R1,B            ;SAVE START ADDR
013C 90018E          178              MOV     DPTR,#ENDMES
013F 12031D          179              CALL    SPMES
0142 12029A          180              CALL    HEXIN
0145 8682            181              MOV     DPL,R0          ;BRING BACK START ADDR
0147 8983            182              MOV     DPH,R1
0149 F8              183              MOV     R0,A            ;SAVE END ADDR
014A A9F0            184              MOV     R1,B
014C 120203          185      DCLOP1: CALL    NEWLIN
014F E583            186              MOV     A,DPH
0151 12026B          187              CALL    HEXOUT
0154 E582            188              MOV     A,DPL
0156 12026B          189              CALL    HEXOUT          ;OUTPUT ADDR
0159 742D            190              MOV     A,#'-'
015B 1201F1          191              CALL    SPOUT
015E 7A10            192              MOV     R2,#10H         ;LOAD COUNT OF BYTES OUTPUT
0160 7420            193      DCLOP2: MOV     A,#' '
0162 1201F1          194              CALL    SPOUT
0165 30980B          195              JNB     RI,DC2
0168 1201F9          196              CALL    SPIN            ;SEE WHAT WAS TYPED
016B B40302          197              CJNE    A,#03,DC3
016E 0137            198              JMP     CMD             ;EXIT IF CONTROL C
0170 1201F9          199      DC3:    CALL    SPIN            ;IF NOT CONT. C WAIT FOR ANYTHING
0173 E4              200      DC2:    CLR     A
0174 93              201              MOVC    A,@A+DPTR       ;GET CODE BYTE
0175 12026B          202              CALL    HEXOUT          ;OUTPUT IT
0178 E9              203              MOV     A,R1
0179 B58306          204              CJNE    A,DPH,DC1       ;DOES ADDR=END ADDR
017C E8              205              MOV     A,R0
017D B58202          206              CJNE    A,DPL,DC1
0180 0137            207              JMP     CMD             ;IF SO EXIT
0182 A3              208      DC1:    INC     DPTR
0183 DADB            209              DJNZ    R2,DCLOP2
0185 80C5            210              JMP     DCLOP1          ;IF COUNT=0 START NEW LINE
                     211
0187 2066726F        212      STAMES: DB ' FROM ',00
018B 6D20
018D 00
018E 20746F20        213      ENDMES: DB ' TO ',00
```

```
0192 00
                        214
                        215
                        216     ;THIS COMMAND WILL EXEC. USER PROGRAM STARTING AT INPUT ADDR
                        217
0193 12029A             218     GO:     CALL    HEXIN           ;GET ADDR TO JUMP TO
0196 120203             219             CALL    NEWLIN
0199 F582               220             MOV     DPL,A           ;DPTR <= ADDR
019B 85F083             221             MOV     DPH,B
019E E4                 222             CLR     A
019F 73                 223             JMP     @A+DPTR         ;PC <= (DPTR)
                        224
                        225     ;THIS COMMAND CHANGES THE BAUD RATE (SEE NOTE BELOW)
                        226
01A0 12029A             227     BAUD:   CALL    HEXIN           ;IF SO BRING IN BAUD
01A3 B40700             228             CJNE    A,#07,$+3
01A6 4002               229             JC      BAUDOK          ;DONT EXIT IF GOOD BAUD INPUT
01A8 0137               230             JMP     CMD
01AA 9001B2             231     BAUDOK: MOV     DPTR,#BAUDTBL   ;POINT TO LOOK UP TABLE
01AD 93                 232             MOVC    A,@A+DPTR       ;GET RELOAD VALUE FROM TABLE
01AE F58D               233             MOV     TH1,A           ;SET THE RELOAD REG.
01B0 0137               234             JMP     CMD
                        235
                        236
                        237     ;THE FOLLOWING IS A LOOK UP TABLE TO GET
                        238     ;THE AUTO-RELOAD VALUE FOR TIMER 1 FOR
                        239     ;SETTING THE BAUD RATE.
                        240     ; TO SET THE RATE HIT B AND 0 - 6
                        241     ;       0=4800
                        242     ;       1=2400
                        243     ;       2=1200
                        244     ;       3=600
                        245     ;       4=300
                        246     ;       5=150
                        247     ;       6=110
                        248
01B2 FF                 249     BAUDTBL:        DB      -1,-2,-4,-8,-16,-32,-44
01B3 FE
01B4 FC
01B5 F8
01B6 F0
01B7 E0
01B8 D4
                        250
                        251
                        252
----                    253             DSEG AT 08
                        254
0008                    255     TEMP:   DS      01
0009                    256     NAMH:   DS      01
000A                    257     NAML:   DS      01
                        258
                        259
                        260     ;THE FOLLOWING ROUTINES ARE USED BY THE MONITOR, BUT
                        261     ;MANY MAY BE CALLED BY USER PROGARMS.
```

```
                        262     ;THOSE THAT MAY ARE MARKED WITH *'S.
                        263
                        264     ;PORT 1 IS DEFINED AS FOLLOWS:
                        265     ;  P1    7   6   5   4    3    2   1   0
                        266     ;        SBR,SBB,ACK,OREQ,RLBK,LBLK,NC,NC
                        267     ; I/O    0   1   1   0    0    0   1   1
                        268
                        269     PUBLIC WRMES,RDMES,SPIN,SPOUT,ECHO,BLOCK,OUTMES,INMES
                        270     PUBLIC NEWLIN,CROUT,GETSB,RELSB,HEXOUT,HEXIN,SPMES,NAM?
                        271
                        272
  0073                  273     OREQ    EQU     73H     ;THE WORD FOR OPEN REQUEST
                        274                             ;AND KEEP INPUT PINS INPUTS
  00EF                  275     REL     EQU     0EFH    ;NOT OF ABOVE
  0095                  276     ACK     BIT     P1.5    ;DEFINE ACK BIT
                        277
----                    278             CSEG
                        279
                        280     ;INITIALIZATION ROUTINE
                        281
                        282     INIT:
01B9 759063             283             MOV     P1,#63H         ;SETUP DIRECTION OF P1
01BC 75A000             284             MOV     P2,#00H         ;MAKE PORT 2 LOW
                        285                                     ;WHEN MOVX IS NOT PERFORMED
01BF 750B00             286             MOV     BLOCK,#00       ;CLEAR BLK'S
01C2 22                 287             RET
                        288
                        289     ;***** WRMES - WRITE MESSAGE *****
                        290
                        291     ;THIS ROUTINE WRITES THE MESSAGE IN ACC TO (DPTR)
                        292
                        293
01C3 F50C               294     WRMES:  MOV     OUTMES,A
01C5 E50B               295     GETBUS: MOV     A,BLOCK
01C7 4473               296             ORL     A,#OREQ         ;PORT1 <= DIR+BLK+OREQ
01C9 F590               297             MOV     P1,A            ;OUTPUT THE REQUEST
01CB 2095FD             298     ACKLOP: JB      ACK,$           ;WAIT FOR ACK TO GO LOW
01CE E50C               299     GOTBUS: MOV     A,OUTMES
01D0 F0                 300             MOVX    @DPTR,A         ;PUT (OUTMES) IN MESSADDR
                        301
                        302
01D1 5390EF             303     RELBUS: ANL     P1,#REL         ;CLEAR OREQ BIT
01D4 22                 304             RET
                        305
                        306     ;***** RDMES - READ MESSAGE *****
                        307
                        308     ;THIS ROUTINE READ MESSAGES FROM MAILBOXS OR I/O DEVICES ADDRESSED
                        309     ;BY THE (DPTR) AND LEAVES IT IN ACC AND INMES.
                        310
                        311
01D5 C083               312     RDMES:  PUSH    DPH
01D7 438340             313             ORL     DPH,#40H        ;MAKE IT PRIVATE
01DA E0                 314             MOVX    A,@DPTR
01DB F50D               315             MOV     INMES,A         ;STORE MESSAGE IN INMES
01DD D083               316             POP     DPH             ;BESURE DPTR IS NOT CHANGED
```

```
010F 22          317             RET
                 318
----             319             DSEG
000B             320     BLOCK:  DS      01      ;HOLDS VALUE OF RBLK AND LBLK
                 321                             ;       08 = RIGHT BLOCKED
                 322                             ;       04 = LEFT BLOCKED
                 323                             ;       0C = BOTH BLOCKED
                 324                             ;       00 = NONE BLOCKED
000C             325     OUTMES: DS      01      ;HOLDS MESSAGE TO GO OUT
000D             326     INMES:  DS      01      ;RECEIVES MESSAGE COMEING IN
                 327
                 328
----             329                 CSEG
                 330
                 331     ;INITALIZE THE SERIAL PORT AND BAUD TIMER
                 332
01E0 759852      333     SPINIT: MOV     SCON,#01010010B ;SET PORT MODE 2 AND SET XMIT READY
                 334
01E3 758960      335     TIINIT: MOV     TMOD,#01100000B ;SET TIMER 1 TO AUTO RELOAD
                 336                                     ;AND FOR EXTERNAL REFERENCE
01E6 758DFF      337             MOV     TH1,#-1         ;SET TO DIVIDE BY 1 (4800 BAUD)
01E9 D28E        338             SETB    TR1             ;START TIMER 1
01EB 22          339             RET
                 340
                 341     ;***** - ECHO - *****
                 342
                 343     ;THIS ROUTINE WAITS FOR A CHARACTER FROM THE TERMINAL,
                 344     ;SAVES IT IN ACC AND ECHOS IT BACK TO THE TERINAL
                 345
01EC 31F9        346     ECHO:   ACALL   SPIN
01EE 31F1        347             ACALL   SPOUT
01F0 22          348             RET
                 349
                 350     ;***** SPOUT - SERIAL PORT OUT - *****
                 351
                 352     ;OUTPUT A CHARACTER IN ACC TO THE TERMINAL
                 353
01F1 3099FD      354     SPOUT:  JNB     TI,$            ;WAIT FOR LAST CHARACTER TO GET OUT
01F4 C299        355             CLR     TI              ;CLEAR TRANSMITTION COMPLETE FLAG
01F6 F599        356             MOV     SBUF,A
01F8 22          357             RET
                 358
                 359     ;***** SPIN - SERIAL PORT IN - *****
                 360
                 361     ;THIS ROUTINE WAITS FOR A CHARACTER FROM THE TERMINAL
                 362     ;AND PUTS IN THE ACC
                 363
01F9 3098FD      364     SPIN:   JNB     RI,$            ;WAIT FOR KEY TO BE HIT
01FC C298        365             CLR     RI              ;CLEAR FLAG
01FE E599        366             MOV     A,SBUF
0200 C2E7        367             CLR     ACC.7           ;REMOVE PARITY
0002 22          368             RET
                 369
                 370     ;***** NEWLIN - NEW LINE *****
                 371
```

```
                        372     ;THIS ROUTINE OUTPUTS A CR,LF TO THE TERMINAL
                        373
0203 C0E0               374     NEWLIN: PUSH    ACC
0205 740D               375             MOV     A,#0DH          ;OUT CR,LF
0207 31F1               376             CALL    SPOUT
0209 740A               377             MOV     A,#0AH
020B 31F1               378             CALL    SPOUT
020D D0E0               379             POP     ACC
020F 22                 380             RET
                        381
                        382     ;***** CROUT - CRT OUT *****
                        383
                        384     ;THIS GETS THE SERIAL BUS AND OUTPUTS THE HEX VALUE WITH
                        385     ;THE NAME OF THE NODE IT CAME FROM TO THE TERMINAL
                        386
                        387
0210 C0E0               388     CROUT:  PUSH    ACC
0212 C0E0               389             PUSH    ACC
0214 12022D             390             CALL    GETSB           ;GET THE BUS
0217 7400      F        391             MOV     A,#NAM
0219 12026B             392             CALL    HEXOUT
021C 742D               393             MOV     A,#'-'
021E 31F1               394             CALL    SPOUT           ;OUTPUT PROMPT TO ID THE NODE
0220 D0E0               395             POP     ACC
0222 12026B             396             CALL    HEXOUT          ;OUTPUT THE VALUES AS A TWO DIGIT HEX
0225 5103               397             CALL    NEWLIN
0227 12023E             398             CALL    RELSB           ;RELEASE THE BUS
022A D0E0               399             POP     ACC
022C 22                 400             RET
                        401
                        402     ;***** GETSB - GET SERIAL BUS *****
                        403
                        404     ;THIS CAPTURES THE SERIAL BUS
                        405
022D C002               406     GETSB:  PUSH    02              ;SAVE R2
022F 209709             407             JB      SBR,GSBEX       ;EXIT IF YOU ALREADY HAVE IT
0232 7A7F               408     GSBLOP: MOV     R2,#7FH         ;SET FOR A DELAY
0234 DAFE               409             DJNZ    R2,$            ;DELAY
0236 3096F9             410             JNB     SBR,GSBLOP      ;TRY TO GET IF TRY AGAIN LATER
0239 D297               411             SETB    SBR             ;IF IT WAS FREE TAKE IT
023B D002               412     GSBEX:  POP     02
023D 22                 413             RET
                        414     ;***** RELSB - RELEASE THE SERIAL BUS *****
                        415
                        416     ;THIS ROUTINE GIVES UP THE SERIAL BUS
                        417
023E 3099FD             418     RELSB:  JNB     TI,$    ;WAIT TO RELEASE SBUS UNTIL ANY CHAR. IS OUT
0241 C297               419             CLR     SBR
0243 22                 420             RET
                        421
                        422     ;***** NAM? - IS NAME CALLED? *****
                        423
                        424     ;THIS TESTS IF THE CONTROL SEQUENCE TOOK PLACE AND IF THIS NODE'S
                        425     ;NAME WAS CALLED. IF SO IT JUMPS TO CMD, IF NOT IT RETURNS.
                        426
```

```
                        427
0244 309803             428     NAM?:   JNB     RI,NEXIT        ;SEE IF SBUF IS FULL
0247 12024B             429             CALL    ISNAM
024A 22                 430     NEXIT:  RET
                        431
024B C0E0               432     ISNAM:  PUSH    ACC
024D 31F9               433             CALL    SPIN            ;IF SO GET IT
024F B40E16             434             CJNE    A,#0EH,ISEXT    ;RET IF NOT CONTROL N
0252 31F9               435             CALL    SPIN            ;IF SO GET NAME TO FOLLOW
0254 B50911             436             CJNE    A,NAMH,ISEXT    ;RET IF NOT OUR NAME
0257 31F9               437             CALL    SPIN            ;IF SO GO TO CMD
0259 B50A0C             438             CJNE    A,NAML,ISEXT
025C 512D               439             CALL    GETSB
025E 5103               440             CALL    NEWLIN
0260 D0E0               441             POP     ACC             ;BUT STACK MUST BE FIXED
0262 D0E0               442             POP     ACC
0264 D0E0               443             POP     ACC
0266 0137               444             JMP     CMD
0268 D0E0               445     ISEXT:  POP     ACC             ;ACC IS NOT CHANGED IF RETURN TO
026A 22                 446             RET                     ;USER PROGRAM
                        447
                        448     ;***** HEXOUT - HEX VALUE OUT *****
                        449
                        450     ;CONVERT HEX VALUE IN ACC TO TWO ASCII CHAR'S.
                        451     ;AND OUTPUT THEM TO THE TERMINAL.
                        452
026B C082               453     HEXOUT: PUSH    DPL
026D C083               454             PUSH    DPH
026F C0E0               455             PUSH    ACC
0271 C0E0               456             PUSH    ACC
0273 54F0               457             ANL     A,#0F0H         ;REMOVE LOW DIGIT
0275 C4                 458             SWAP    A
0276 90028A             459             MOV     DPTR,#HEXTBL
0279 93                 460             MOVC    A,@A+DPTR       ;GET ASCII CHAR. FROM TABLE
027A 31F1               461             CALL    SPOUT           ;OUTPUT THAT CHAR.
027C D0E0               462             POP     ACC
027E 540F               463             ANL     A,#0FH          ;REMOVE UPPER DIGIT
0280 93                 464             MOVC    A,@A+DPTR       ;GET NEXT ASCII CHAR.
0281 31F1               465             CALL    SPOUT
0283 D0E0               466             POP     ACC
0285 D083               467             POP     DPH
0287 D082               468             POP     DPL
0289 22                 469             RET
                        470
028A 30313233           471     HEXTBL: DB      '0123456789ABCDEF'
028E 34353637
0292 38394142
0296 43444546
                        472
                        473     ;***** HEXIN - HEX VALUE IN *****
                        474
                        475     ;TAKE IN <= 4 CHAR'S (LEADING 0 ARE FILLED IN) FROM TERM AND CONVERT TO HEX
                        476     ;RESULT IS LEFT IN ACC (LOWER BYTE) AND B (UPPER BYTE)
                        477
029A C001               478     HEXIN:  PUSH    01              ;SAVE R1
```

```
02BC C002          479              PUSH    02              ;SAVE R2
029E C003          480              PUSH    03              ;SAVE R3
02AD 7900          481              MOV     R1,#00
02A2 75F000        482              MOV     B,#00           ;CLR B (UPPER BYTE TO RETURN)
02A5 7A00          483              MOV     R2,#00
02A7 C001          484              PUSH    01              ;PUT LEADING ZERO ON STACK
02A9 31F9          485  HEXINL:     CALL    SPIN
02AB B4080C        486              CJNE    A,#08,HEXIN1    ;WAS THE INPUT A BACK SPACE
02AE B90002        487              CJNE    R1,#00,HEXRUB
02B1 80F6          488              JMP     HEXINL          ;GET NEW INPUT IF NOTHING TO RUBOUT
02B3 31F1          489  HEXRUB:     CALL    SPOUT           ;OUTPUT THE BACK SPACE
02B5 19            490              DEC     R1
02B6 D0E0          491              POP     ACC             ;AND FORGET LAST INPUT
02B8 80EF          492              JMP     HEXINL
02BA B90403        493  HEXIN1:     CJNE    R1,#04,HEXIN2   ;QUIT AFTER 4 INPUTS
02BD 0202FC        494              JMP     HXDON1
02C0 B40D03        495  HEXIN2:     CJNE    A,#0DH,HEXIN3   ;QUIT IF CR
02C3 0202F6        496              JMP     HXDONE
02C6 B42003        497  HEXIN3:     CJNE    A,#' ',HEXIN4   ;ALSO QUIT IF SP
02C9 0202F6        498              JMP     HXDONE
02CC FB            499  HEXIN4:     MOV     R3,A
02CD 54F0          500              ANL     A,#0F0H
02CF B4300B        501              CJNE    A,#30H,FOUR?    ;TEST IF IT IS A NUMBER
02D2 EB            502              MOV     A,R3
02D3 B43A00        503              CJNE    A,#':',$+3      ;IS ACC > '9' ?
02D6 50D1          504              JNC     HEXINL          ;IF NOT TRY AGAIN
02D8 31F1          505              CALL    SPOUT           ;OUTPUT THE LEGAL NUMBER
02DA 0202EF        506              JMP     HXSAVE
                   507
02DD B440C9        508  FOUR?:     CJNE    A,#40H,HEXINL   ;TEST IF IT IS A-F
02E0 EB            509              MOV     A,R3
02E1 B44002        510              CJNE    A,#'@',HEXCON   ;TRY AGAIN IF '@'
02E4 80C3          511              JMP     HEXINL
02E6 B44700        512  HEXCON:     CJNE    A,#'G',$+3
02E9 50BE          513              JNC     HEXINL          ;JMP IF >= G
02EB 31F1          514              CALL    SPOUT           ;OUTPUT LEGAL LETTER
02ED 2409          515              ADD     A,#09           ;ADD OFSET FOR LETTERS
                   516
02EF 540F          517  HXSAVE:     ANL     A,#0FH          ;REMOVE ANY GARBAGE
02F1 C0E0          518              PUSH    ACC             ;SAVE ON STACK
02F3 09            519              INC     R1              ;INC COUNT
02F4 80B3          520              JMP     HEXINL
                   521
02F6 B90003        522  HXDONE:     CJNE    R1,#00,HXDON1
02F9 C2030F        523              JMP     HXEXIT          ;EXIT IF NOTHING DONE (0000 IS RETURNED)
02FC D002          524  HXDON1:     POP     02              ;GET SAVED INPUT
02FE D0E0          525              POP     ACC
0300 C4            526              SWAP    A               ;MOV UPPER DIGIT TO UPPER NIBBLE
0301 4202          527              ORL     02,A            ;SAVE BYTE IN R2
0303 B90300        528              CJNE    R1,#03,$+3
0306 4007          529              JC      HXEXIT          ;DONE IF ONLY ONE BYTE INPUT
0308 D0F0          530              POP     B               ;SAME BUT PUT IN B
030A D0E0          531              POP     ACC
030C C4            532              SWAP    A
030D 42F0          533              ORL     B,A
```

```
                        534
030F E9                 535     HXEXIT: MOV     A,R1
0310 20E002             536             JB      ACC.0,HXEXT1
0313 D001               537             POP     01              ;IF THE COUNT IS EVEN WE MUST POP
                        538                                     ;OFF THE LEADING ZERO
0315 EA                 539     HXEXT1: MOV     A,R2            ;PUT LOWER BYTE IN A
0316 D003               540             POP     03
0318 D002               541             POP     02
031A D001               542             POP     01
031C 22                 543             RET
                        544
                        545     ;***** SPMES - SERIAL PORT MESSAGE *****
                        546
                        547     ;THIS ROUTINE OUTPUTS A STRING TO THE TERMNIAL
                        548     ;  THE DPTR HOLD THE ADDRESS OF THE FIRST CHARACTER
                        549     ;OF THE STRING AND THE LAST BYTE MUST BE 00
                        550
031D C0E0               551     SPMES:  PUSH    ACC
031F C2D1               552             CLR     PSW.1           ;CLR THE USER DEFINABLE FLAG
0321 209704             553             JB      SBR,SPMLOP
0324 D2D1               554             SETB    PSW.1           ;IF WE DONT HAVE THE SBUS SET FLAG
0326 512D               555             CALL    GETSB           ;AND GET THE SBUS
0328 E4                 556     SPMLOP: CLR     A
0329 93                 557             MOVC    A,@A+DPTR
032A 6005               558             JZ      SPMEXT
032C 31F1               559             CALL    SPOUT
032E A3                 560             INC     DPTR
032F 80F7               561             JMP     SPMLOP
0331 30D102             562     SPMEXT: JNB     PSW.1,SPMPOP    ;IF WE HAD THE SBUS DONT RELEASE IT
0334 513E               563             CALL    RELSB
0336 D0E0               564     SPMPOP: POP     ACC
0338 22                 565             RET
                        566
                        567             END
```

| NAME | TYPE | VALUE | ATTRIBUTES |
|---|---|---|---|
| ACC. . . . | D ADDR | 00E0H | A |
| ACK. . . . | B ADDR | 0090H.5 | A |
| ACKLOP . . | C ADDR | 01CBH | A |
| B. . . . . | D ADDR | 00F0H | A |
| BAUD . . . | C ADDR | 01A0H | A |
| BAUDOK . . | C ADDR | 01AAH | A |
| BAUDTBL. . | C ADDR | 01B2H | A |
| BLKRTN . . | C ADDR | 0096H | A |
| BLOCK. . . | D ADDR | 0008H | A PUB |
| CMD. . . . | C ADDR | 0037H | A PUB |
| CMDL . . . | C ADDR | 0052H | A |
| CMDTBL . . | C ADDR | 007EH | A |
| CONT . . . | C ADDR | 005DH | A |
| CROUT. . . | C ADDR | 0210H | A PUB |
| DC1. . . . | C ADDR | 0182H | A |
| DC2. . . . | C ADDR | 0173H | A |
| DC3. . . . | C ADDR | 0170H | A |
| DCLOP1 . . | C ADDR | 014CH | A |
| DCLOP2 . . | C ADDR | 0160H | A |
| DISCOD . . | C ADDR | 0130H | A |
| DPH. . . . | D ADDR | 0083H | A |
| DPL. . . . | D ADDR | 0082H | A |
| ECHO . . . | C ADDR | 01ECH | A PUB |
| ENDMES . . | C ADDR | 018EH | A |
| ERROR. . . | C ADDR | 0074H | A |
| FOUR?. . . | C ADDR | 02DDH | A |
| GETBUS . . | C ADDR | 01C5H | A |
| GETSB. . . | C ADDR | 022DH | A PUB |
| GO . . . . | C ADDR | 0193H | A |
| GOTBUS . . | C ADDR | 01CEH | A |
| GSBEX. . . | C ADDR | 023BH | A |
| GSBLOP . . | C ADDR | 0232H | A |
| HEXCON . . | C ADDR | 02E6H | A |
| HEXIN. . . | C ADDR | 029AH | A PUB |
| HEXIN1 . . | C ADDR | 02BAH | A |
| HEXIN2 . . | C ADDR | 02C0H | A |
| HEXIN3 . . | C ADDR | 02C6H | A |
| HEXIN4 . . | C ADDR | 02CCH | A |
| HEXINL . . | C ADDR | 02A9H | A |
| HEXOUT . . | C ADDR | 026BH | A PUB |
| HEXRUB . . | C ADDR | 02B3H | A |
| HEXTBL . . | C ADDR | 028AH | A |
| HXDON1 . . | C ADDR | 02FCH | A |
| HXDONE . . | C ADDR | 02F6H | A |
| HXEXIT . . | C ADDR | 030FH | A |
| HXEXT1 . . | C ADDR | 0315H | A |
| HXSAVE . . | C ADDR | 02EFH | A |
| INIT . . . | C ADDR | 01B9H | A |
| INMES. . . | D ADDR | 000DH | A PUB |
| ISEXT. . . | C ADDR | 0268H | A |
| ISNAM. . . | C ADDR | 024BH | A |

```
LEFTB. . .  C ADDR   00A4H   A
LEFTU. . .  C ADDR   00B7H   A
NAM. . . .    NUMB   ----      EXT
NAM? . . .  C ADDR   0244H   A PUB
NAMH . . .  D ADDR   0009H   A
NAML . . .  D ADDR   000AH   A
NEWLIN . .  C ADDR   0203H   A PUB
NEXIT. . .  C ADDR   024AH   A
NWAIT. . .  C ADDR   002BH   A
OREQ . . .    NUMB   0073H   A
OUTMES . .  D ADDR   000CH   A PUB
P1 . . . .  D ADDR   0090H   A
P2 . . . .  D ADDR   00A0H   A
PSW. . . .  D ADDR   00D0H   A
RDMES. . .  C ADDR   01D5H   A PUB
REL. . . .    NUMB   00EFH   A
RELBUS . .  C ADDR   01D1H   A
RELSB. . .  C ADDR   023EH   A PUB
RI . . . .  B ADDR   0098H.0 A
RIGHTB . .  C ADDR   009FH   A
RIGHTU . .  C ADDR   00B2H   A
SBB. . . .  B ADDR   0090H.6 A
SBN0 . . .  C ADDR   00D8H   A
SBM1 . . .  C ADDR   00DAH   A
SBM2 . . .  C ADDR   00E2H   A
SBM25. . .  C ADDR   00E8H   A
SBM3 . . .  C ADDR   00F7H   A
SBM4 . . .  C ADDR   010CH   A
SBM5 . . .  C ADDR   0113H   A
SBM6 . . .  C ADDR   011FH   A
SBM7 . . .  C ADDR   0120H   A
SBR. . . .  B ADDR   0090H.7 A
SBUF . . .  D ADDR   0099H   A
SCON . . .  D ADDR   0098H   A
SMNEXT . .  C ADDR   0125H   A
SMOK . . .  C ADDR   00C8H   A
SP . . . .  D ADDR   0081H   A
SPIN . . .  C ADDR   01F9H   A PUB
SPINIT . .  C ADDR   01E0H   A
SPMES. . .  C ADDR   031DH   A PUB
SPMEXT . .  C ADDR   0331H   A
SPMLOP . .  C ADDR   0328H   A
SPMPOP . .  C ADDR   0336H   A
SPOUT. . .  C ADDR   01F1H   A PUB
STAMES . .  C ADDR   0187H   A
START. . .  C ADDR   0010H   A
SUBMEM . .  C ADDR   00BCH   A
SX?. . . .  C ADDR   00C5H   A
TBLEND . .  C ADDR   0096H   A
TEMP . . .  D ADDR   0008H   A
TH1. . . .  D ADDR   008DH   A
TI . . . .  B ADDR   0098H.1 A
TIINIT . .  C ADDR   01E3H   A
TMOD . . .  D ADDR   0089H   A
TR1. . . .  B ADDR   0088H.6 A
```

```
UNBLK. . .   C ADDR     00A9H   A
USER . . .   C ADDR     ----      EXT
WRMES. . .   C ADDR     01C3H   A PUB

REGISTER BANK(S) USED: 0, TARGET MACHINE(S): 8051

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

# APPENDIX B

## DEMONSTRATION SOFTWARE LISTING

```
LOC  OBJ                  LINE     SOURCE

                             1     $TITLE(USER PROG FOR PHOTO-TRANSISTER POLLING (SEEKER) 17-JULY-82)
                             2
  6000                       3     IOBIT   EQU     6000H   ;INDICATES I/O OPERATION
  1002                       4     SPDMAL  EQU     1002H   ;MAILBOX FOR SPEED
  1005                       5     PANMAL  EQU     1005H   ;MAILBOX FOR PAIN VALUE
  0000                       6     ZERO    BIT     00      ;SET IF HAND EYE IS ON
  00B2                       7     EOC     BIT     P3.2    ;END OF CONVERSION SENT TO PROCESSOR
  0002                       8     NAM     EQU     02      ;ID OF PARTICULAR PROCESSOR
                             9
                            10     ; REQUIRED UTILITY PROGRAMS AND DATA LOCATIONS
                            11     EXTRN   CODE    (WRMES,CROUT,SPMES,SPIN,HEXIN,NEWLIN,NAM?,RELSB)
                            12     EXTRN   DATA    (OUTMES)
                            13     PUBLIC  USER,NAM
                            14
                            15     USERS   SEGMENT CODE
----                        16             RSEG    USERS
                            17
0000 900000     F           18     USER:   MOV     DPTR,#MESAGE    ;REQUEST DEFAULT SPEED CONSTANT
0003 120000     F           19             CALL    SPMES
0006 120000     F           20             CALL    HEXIN
0009 120000     F           21             CALL    NEWLIN
000C 120000     F           22             CALL    RELSB           ;RELEASE SERIAL BUS
000F 540F                   23             ANL     A,#0FH          ;SPEED IS ONLY 4 BITS
0011 F500       F           24             MOV     SPDC,A          ;STORE SPEED INPUT
                            25
                            26     ; MAIN : PRIMARY PROGRAM LOOP
0013 900000     F           27     MAIN:   MOV     DPTR,#MES1      ;TELL USER HE IS IN MAIN
0016 120000     F           28             CALL    SPMES
0019 120000     F           29             CALL    SCAN            ;SCAN DETECTORS
001C E500       F           30             MOV     A,PTABLE        ;PTABLE HOLDS NUMBER ON
001E 20E014                 31             JB      ACC.0,ONEON     ;IF ONE ON, GOTO ONEON
0021 20E10E                 32             JB      ACC.1,JTWOON    ;IF TWO ON, GOTO TWOON
                            33
                            34     ; ZON : FIND HAND BY MOVING HAND AND WATCHING DETECTORS
0024 900000     F           35     ZON:    MOV     DPTR,#MES0      ;ELSE ZERO ON, LET USER KNOW
0027 120000     F           36             CALL    SPMES
002A 850000     F           37             MOV     SPEED,SPDC      ;MOVE HAND AT DEFAULT SPEED
002D 120000     F           38             CALL    SNDSPD
0030 80E1                   39             JMP     MAIN
                            40
0032 020000     F           41     JTWOON: JMP     TWOON           ;TOO FAR FOR REL JMP
                            42
                            43     ; ONEON : LOOP WHEN SCAN SAYS ONLY ONE DETECTOR ON
0035 900000     F           44     ONEON:  MOV     DPTR,#MES2      ;LET USER KNOW ONE ON
0038 120000     F           45             CALL    SPMES
003B 200060                 46             JB      ZERO,HANDON     ; IF THE ONE IS THE HAND DETECTOR JMP
003E 850000     F           47     ONE1:   MOV     TEMP,PTABLE+1   ;SAVE POSITION OF ONE ON
0041 850000     F           48     ONE2:   MOV     SPEED,SPDC      ;MOVE HAND TO TRY TO FIND IT
0044 120000     F           49             CALL    SNDSPD
0047 120000     F           50             CALL    SCAN            ;WATCH DETECTORS
```

```
004A E500     F      51              MOV     A,PTABLE        ;PTABLE HOLDS NUMBER OF DET. ON
004C 20E158          52              JB      ACC.1,TWOON     ;IF NOW TWO THEN ONE IS FLASHLIGHT(FL)
004F 30E0EF          53              JNB     ACC.0,ONE2      ;IF NONE THEN HAND IS BETWEEN DET., REPEAT
0052 E500     F      54              MOV     A,PTABLE+1      ;ELSE ONE ON
0054 B50002   F      55              CJNE    A,TEMP,ONE3     ;IS IT SAME ONE?
0057 80E8            56              JMP     ONE2            ;YES, SO REPEAT
0059 F500     F      57    ONE3:     MOV     TEMP,A          ;NOT SAME, SO HAND IS FOUND
005B 750000   F      58    ONE4:     MOV     SPEED,#00       ;STOP HAND
005E 120000   F      59              CALL    SNDSPD
0061 120000   F      60              CALL    SCAN            ;WATCH DETECTORS
0064 E500     F      61              MOV     A,PTABLE
0066 20E005          62              JB      ACC.0,ONE5      ;IF ONE ON JMP TO ONE5
0069 20E125          63              JB      ACC.1,ONE7      ;IF TWO THEN FL HAS APPEARED
006C 80ED            64              JMP     ONE4            ;ELSE HAND MUST HAVE OVERSHOT DET., STILL OK
                     65
006E E500     F      66    ONE5:     MOV     A,PTABLE+1      ;COMPARE TO ONE THAT IS HAND
0070 B50002   F      67              CJNE    A,TEMP,ONE6     ;NOT EQUAL FL PRESENT
0073 80E6            68              JMP     ONE4            ;IF EQUAL, REPEAT
                     69
                     70    ; WHEN HERE THE HAND IS BETWEEN DET. AND A FL HAS APPEARED
                     71    ; YET WE STILL KNOW WHERE THE HAND IS FROM PREVIOUS SCANS
0075 7800     F      72    ONE6:     MOV     R0,#PTABLE+1    ;SET ADDRESS IN R0 OF DEFAULT NON HAND LOC.
0077 400B            73              JC      TEMPGT          ;JMP IF HAND POS. > LIGHT (SET BY CJNE IN ONE5)
0079 7800     F      74              MOV     R0,#PTABLE+2    ;HAND IS NOT IN +2
007B 850000   F      75              MOV     PTABLE+2,PTABLE+1       ;PUT IN INCREASING POS. ORDER
007E 850000   F      76              MOV     PTABLE+1,TEMP   ;TEMP HOLDS HAND POS.
0081 020000   F      77              JMP     ONEEXT
0084 850000   F      78    TEMPGT:   MOV     PTABLE+2,TEMP   ;INCREASING POS. ORDER AND TEMP HOLDS HAND POS.
0087 E500     F      79    ONEEXT:   MOV     A,TEMP          ;EXEC WANTS ACC TO HAVE HAND POS.
0089 7500FF   F      80              MOV     TEMP2,#0FFH     ;REQUIRED BY EXEC.
008C 120000   F      81              CALL    EXEC
008F 8082            82              JMP     MAIN
                     83
                     84    ; WHEN HERE FLASHLIGHT(FL) HAS TURNED ON A DETECTOR(DET.), ALSO
                     85    ; THE HAND IS ON A DET.
0091 E500     F      86    ONE7:     MOV     A,PTABLE+1
0093 B50004   F      87              CJNE    A,TEMP,ONE8     ;FIND WHICH IS THE HAND, HAND POS IS IN'TEMP
0096 7800     F      88              MOV     R0,#PTABLE+2    ;ADD. OF NOT HAND REQUIRED IN R0
0098 80ED            89              JMP     ONEEXT
009A 7800     F      90    ONE8.     MOV     R0,#PTABLE+1    ;ADD. OF NOT HAND REQUIRED IN R0
009C 80E9            91              JMP     ONEEXT
                     92
                     93    ; STOPS HAND WHEN DESIRED TO GO BACK TO MAIN LOOP AFTER STOPPED
009E 750000   F      94    HANDON:   MOV     SPEED,#00
00A1 120000   F      95              CALL    SNDSPD
00A4 020000   F      96              JMP     MAIN
                     97
                     98    ; TWOON : USED WHEN POS. OF HAND NOT KNOWN AND THERE ARE TWO ON
00A7 900000   F      99    TWOON:    MOV     DPTR,#MES3      ;LET USER KNOW TWO ON
00AA 120000   F     100              CALL    SPMES
00AD 2000EE         101              JB      ZERO,HANDON     ;IF HANDEYE IS ON STOP
00B0 850000   F     102              MOV     TEMP1,PTABLE+1  ;SAVE POS. OF THOSE ON
00B3 850000   F     103              MOV     TEMP2,PTABLE+2
00B6 850000   F     104              MOV     SPEED,SPDC      ;MOVE HAND TO TRY TO FIND IT
00B9 120000   F     105              CALL    SNDSPD
```

```
00BC 120000   F     106   TWO:    CALL    SCAN            ;WATCH DETECTORS
00BF 20000C         107           JB      ZERO,HANDON     ;IF HAND CAME ON STOP
00C2 E500     F     108           MOV     A,PTABLE
00C4 20E0F5         109           JB      ACC.0,TWO       ;IF ONLY ONE ON, HAND IS BETWEEN DET.
00C7 30E115         110           JNB     ACC.1,TWZON     ;IF BOTH GO OFF, FL IS GONE AND HAND
                    111                                   ;IS BETWEEN DET.
00CA E500     F     112           MOV     A,PTABLE+1      ;TWO ON
●0CC 7800     F     113           MOV     R0,#PTABLE+2    ;ADD. OF NOT HAND SHOULD BE PUT IN R0
00CE B50008   F     114           CJNE    A,TEMP1,TW1     ;FIND WHICH IS HAND(THE ONE THAT MOVED)
00D1 18             115           DEC     R0              ;HAND IS NOT +1, SO PUT +1 IN R0
00D2 E500     F     116           MOV     A,PTABLE+2      ;PUT HAND POS IN ACC
00D4 B50002   F     117           CJNE    A,TEMP2,TW1     ;CHECK IF NEITHER MOVED
00D7 80E3           118           JMP     TWO             ;IF NEITHER MOVED KEEP LOOKING
00D9 120000   F     119   TW1:    CALL    EXEC
00DC 020000   F     120   TWEXIT: JMP     MAIN
00DF 020000   F     121   TWZON:  JMP     ZON     ;TOO FAR THE ABOVE REL. JMP
                    122
                    123
                    124   ; SNDSPD : SENDS DESIRED SPEED TO THE MOTORS
00E2 901002         125   SNDSPD: MOV     DPTR,#SPDMAL    ;SET UP POINTER
00E5 E500     F     126           MOV     A,SPEED
00E7 4500     F     127           ORL     A,DIR           ;SET CORRECT DIRECTION
00E9 F500     F     128           MOV     OUTMES,A
00EB 120000   F     129           CALL    WRMES           ;SEND SPEED
00EE 22             130           RET
                    131
                    132
                    133   ; SCAN : LOOKS AT ALL THE DETECTORS AND RECORDS THE POS OF THE
                    134   ; ONES THAT ARE ON UP TO TWO BEING ON. IF THE HAND DET. IS ON
                    135   ; THEN IT SETS A FLAG AND RETURNS. THIS ROUTINE ALSO SENDS THE
                    136   ; DESIRED VALUE TO THE PAIN PROCESSOR
00EF C200           137   SCAN:   CLR     ZERO            ;CLEAR HAND ON FLAG
00F1 750000   F     138           MOV     PTABLE,#0       ;SET NUMBER ON TO ZERO
00F4 7800     F     139           MOV     R0,#PTABLE+1    ;SET UP POINTER TO FIRST STORAGE POS
00F6 906000         140           MOV     DPTR,#10BIT     ;SET UP POINTER TO REQ I/O
00F9 F0             141           MOVX    @DPTR,A         ;START CONVERSION(WRITES GARBAGE)
00FA 20B2FD         142           JB      EOC,$           ;WAIT ON EOC TO GO LOW
00FD 30B2FD         143           JNB     EOC,$           ;WHEN EOC COMES HIGH CONV. COMPLETE
0100 E0             144           MOVX    A,@DPTR         ;READ VALUE
0101 F4             145           CPL     A
01●2 FD             146           MOV     R5,A            ;SAVE COMPLEMENTED VALUE
0103 120000   F     147           CALL    SNDPAN          ;SEND PAIN VALUE
●106 906000         148           MOV     DPTR,#10BIT     ;SNDPAN CHANGES DPTR
0109 300023         149           JNB     ZERO,CHKH       ;SNDPAN HAS SET FLAG IF HAND IS ON
                    150                                   ;HAND IS LOWER NIBBLE OF VALUE SO
                    151                                   ;STILL MUST CHECK UPPER. HAND IS
                    152                                   ;POSITION ZERO
010C 750000   F     153           MOV     OFFSET,#0       ;IF ZERO, RECORD POS AS ON
010F 120000   F     154           CALL    RECPOS
0112 020000   F     155           JMP     ESEND           ;RETURN
0115 A3             156   EYSCAN: INC     DPTR            ;CHECK NEXT SET OF DET.
0116 E582           157           MOV     A,DPL
0118 B41003         158           CJNE    A,#10H,ES1      ;IF LOOKED AT THEM ALL QUIT
011B 020000   F     159           JMP     ESEND
011E 120000   F     160   ES1:    CALL    NAM?            ;IS YOUR USER CALLING
```

```
0121 F0            161             MOVX    @DPTR,A         ;START CONV. (WRITES GARBAGE)
0122 20B2FD        162             JB      EOC,$
0125 30B2FD        163             JNB     EOC,$           ;WAIT ON END OF CONV.
0128 E0            164             MOVX    A,@DPTR         ;READ VALUE
                   165
                   166
0129 F4            167             CPL     A
012A FD            168             MOV     R5,A            ;SAVE COMPLEMENTED VALUE
012B 540C          169             ANL     A,#0CH          ;EACH DET. CONSIDERED ON IF > 0C0H
                   170                                     ;WE HAVE THE UPPER NIBBLE OF TWO DIF
                   171                                     ;FERENT DET. IN EACH READ
012D 600E          172             JZ      LOWON           ;LOWER ON IS ON IF ZERO
012F ED            173     CHKH:   MOV     A,R5            ;RESTORE VALUE IN ACC
0130 54C0          174             ANL     A,#0C0H         ;IS HIGH NIBBLE ON?
0132 70E1          175             JNZ     EYSCAN          ;IF NOT, KEEP SCANNING
0134 750008    F   176     HIGHON: MOV     OFFSET,#08H     ;ASSURE THE RECORDING OF THE RIGHT ADD.
0137 120000    F   177             CALL    RECPOS
013A 4009          178             JC      EYSCAN          ;CARRY IS RESET IF TWO ARE ALREADY FOUND ON
013C 22            179     ESEND:  RET
                   180
013D 750000    F   181     LOWON:  MOV     OFFSET,#0       ;RECORD CORRECT ADD.
0140 120000    F   182             CALL    RECPOS
0143 50F7          183             JNC     ESEND           ;CARRY IS RESET IF TWO HAVE BEEN RECORDED ON
0145 80E8          184             JMP     CHKH
                   185
                   186     ; RECPOS : STORES THE POSITION OF THE ONE THAT WAS FOUND ON IN A TABLE
                   187     ; IT CHECKS TO SEE IF THERE ARE TWO RECORDED, IF SO, IT SETS THE CARRY
                   188     ; BIT BEFORE RETURNING
                   189     ; PTABLE - HOLDS THE COUNT OF THOSE ON
                   190     ; PTABLE+1 (REFERRED TO AS +1) - HOLDS POSITION OF FIRST FOUND ON
                   191     ; PTABLE+2 (+2) - HOLDS POSITION OF SECOND DETECTOR FOUND ON
0147 E582          192     RECPOS: MOV     A,DPL                   ;GET ADDRESS OF ON DET.
0149 30E302        193             JNB     ACC.3,REC1              ;CHECK NEED FOR ADDRESS CORRECTION
014C 2408          194             ADD     A,#08H                  ;CORRECT ADDRESS
014E 2500      F   195     REC1:   ADD     A,OFFSET                ;STANDARD ADDRESS CORRECTION
0150 F6            196             MOV     @R0,A                   ;STORE ADDRESS IN TABLE
0151 08            197             INC     R0                      ;KEEP TRACK OF NEXT LOC. IN TABLE
0152 0500      F   198             INC     PTABLE                  ;INCREMENT COUNT
0154 B80000    F   199             CJNE    R0,#PTABLE+3,$+3        ;CHECK FOR FULL TABLE,SETS CARRY IF FULL
0157 22            200             RET
                   201
                   202     ; SNDPAN : SENDS VALUE ON HAND DET. TO PAIN PROCESSOR
0158 540C          203     SNDPAN: ANL     A,#0CH          ;CHECK OF HAND DET. ON
015A 7002          204             JNZ     PAN1
015C D200          205             SETB    ZERO            ;IF HAND ON, SET POSITION ZERO FLAG
015E ED            206     PAN1:   MOV     A,R5            ;GET VALUE BACK IN ORIGINAL FORM
015F F4            207             CPL     A
0160 540F          208             ANL     A,#0FH
0162 901005        209             MOV     DPTR,#PANMAL    ;SET POINTER TO PAIN MAILBOX
0165 F500      F   210             MOV     OUTMES,A
0167 120000    F   211             CALL    WRMES
016A 22            212             RET
                   213
                   214
                   215     ; EXEC : EXECUTES THE FUNCTION OF MOVING THE HAND IN FRONT OF THE FL
```

```
                        216     ; AFTER THE HAND HAS BEEN FOUND AND THE FL IS ON
016B 900000     F       217     EXEC:   MOV     DPTR,#MES4      ;LET USER KNOW HE IS HERE
016E 120000     F       218             CALL    SPMES
0171 B50003     F       219     EXECLP: CJNE    A,TEMP2,EXC1    ;EXPECT HAND LOC. IN ACC, CHECK IF HAND
                        220                                     ;WENT FROM LOC 1 TO 31 WHILE IN EXEC
0174 E500       F       221             MOV     A,PTABLE+2      ;IF SO, CORRECT EXPECTED VALUES
0176 18                 222             DEC     R0
0177 850000     F       223     EXC1:   MOV     TEMP1,PTABLE+1  ;SAVE VALUES OF THOSE THAT ARE ON
017A 850000     F       224             MOV     TEMP2,PTABLE+2
017D 750000     F       225             MOV     DIR,#00H        ;RESET THE DIRECTION
0180 C3                 226             CLR     C               ;CLEAR CARRY BEFORE SUBTRACT
0181 96                 227             SUBB    A,@R0           ;FIND DISTANCE BETWEEN HAND AND FL
0182 30E702             228             JNB     ACC.7,HGRTR
0185 2420               229             ADD     A,#20H          ;IF VALUE IS < 0 MAKE CORRECTION
0187 B41100             230     HGRTR:  CJNE    A,#11H,$+3
018A 4008               231             JC      SPD1            ;IF FARTHER THAN HALF WAY AROUND,
                        232                                     ;CAL SPEED SO GOES OTHER WAY
                        233                                     ;HAND SHOULD ALWAYS GO THE SHORTEST
                        234                                     ;ROUTE WHEN IN EXEC
018C FD                 235             MOV     R5,A
018D 7420               236             MOV     A,#20H
018F C3                 237             CLR     C
0190 9D                 238             SUBB    A,R5
0191 750010     F       239             MOV     DIR,#10H
0194 F500       F       240     SPD1:   MOV     SPEED,A         ;MOVE HAND AT CALCULATED SPEED
0196 120000     F       241     H1:     CALL    SNDSPD
0199 120000     F       242             CALL    SCAN            ;CHECK DETECTORS
019C 200012             243             JB      ZERO,EXDONE     ;IF HAND DETECTOR ON, THEN DONE
019F E500       F       244             MOV     A,PTABLE
01A1 20E114             245             JB      ACC.1,EX4       ;ELSE IF TWO STILL ON JMP TO EX4
01A4 20E0EF             246             JB      ACC.0,H1        ;IF ONLY ONE ON THEN KEEP MOVING TILL HAND
                        247                                     ;COMES BACK ON
01A7 120000     F       248     EZON:   CALL    SNDSPD          ;IF NONE ON, WAIT ON ONE TO COME BACK ON
                        249                                     ;WHEN HERE, EITHER THE FL IS GONE AND THE
                        250                                     ;HAND IS BETWEEN DET. OR THE HAND BULB IS
                        251                                     ;CASTING A SHADOW OVER THE DET THE FL IS
                        252                                     ;ON AND THE HAND IS BETWEEN DET.  SO THAT
                        253                                     ;WHEN ONE COMES BACK ON, IT WILL EITHER
                        254                                     ;BE THE HAND DET OR THE HAND TURNING A DET ON
01AA 120000     F       255             CALL    SCAN
01AD E500       F       256             MOV     A,PTABLE
01AF 60F6               257             JZ      EZON
                        258
01B1 750000     F       259     EXDONE: MOV     SPEED,#00       ;DONE SO STOP HAND
01B4 120000     F       260             CALL    SNDSPD
01B7 22                 261             RET
                        262
01B8 E500       F       263     EX4:    MOV     A,PTABLE+1      ;FIND THE HAND, ASSUME IT IS THE ONE THAT
                        264                                     ;MOVED
01BA 7800       F       265             MOV     R0,#PTABLE+2    ;SET UP R0 AND ACC FOR EXEC
01BC B500B2     F       266             CJNE    A,TEMP1,EXECLP  ;IF TEMP1 HAS CHANGED, IT COULD BE THAT
                        267                                     ;THE HAND HAS MOVED TO POS 31 FROM 0
01BF 18                 268             DEC     R0              ;IF +1 HAS NOT CHANGED, THEN CHECK +2
01C0 E500       F       269             MOV     A,PTABLE+2
01C2 B500B2     F       270             CJNE    A,TEMP2,EXC1
```

```
01C5 80CF                271             JMP     H1
                         272
                         273
01C7 0A                  274     MESAGE: DB OAH,ODH,'INPUT SPDC-',OOH
01C8 0D
01C9 496E7075
01CD 74205350
01D1 44432D
01D4 0O                  275     MES1:   DB OAH,ODH,'MAIN',OAH,ODH,OOH
01D5 0A
01D6 0D
01D7 4D41494E
01DB 0A
01DC 0D
01DD 00                  276     MES0:   DB OAH,ODH,'ENTERED ZON',OAH,ODH,OOH
01DE 0A
01DF 0D
01E0 456E7465
01E4 72656420
01E8 5A4F4E
01EB 0A
01EC 0D
01ED 00                  277     MES2:   DB OAH,ODH,'ENTERED ONEON',OAH,ODH,OOH
01EE 0A
01EF 0D
01F0 456E7465
01F4 72656420
01F8 4F4E454F
01FC 4E
01FD 0A
01FE 0D
01FF 00                  278     MES3:   DB OAH,ODH,'ENTERED TWOON',OAH,ODH,OOH
0200 0A
0201 0D
0202 456E7465
0206 72656420
020A 54574F4F
020E 4E
020F 0A
0210 0D
0211 00                  279     MES4:   DB OAH,ODH,'ENTERED EXEC',OAH,ODH,OOH
0212 0A
0213 0D
0214 456E7465
0218 72656420
021C 45584543
0220 0A
0221 0D
0222 00
                         280
                         281     EYEVAR  SEGMENT DATA
                         282             RSEG    EYEVAR
----
                         283
0000                     284     TEMP:   DS      01      ;TEMPORARY STORAGE
0001                     285     TEMP1:  DS      01      ;TEMPORARY STORAGE
```

```
LOC  OBJ             LINE     SOURCE

0002                  286     TEMP2:   DS      01      ;TEMPORARY STORAGE
0003                  287     SPEED:   DS      01      ;SPACE TO HOLD CURRENT SPEED

0004                  288     SPDC:    DS      01      ;HOLDS SPEED CONSTANT INPUT BY USER
0005                  289     PTABLE:  DS      03      ;POSITION TABLE
0008                  290     OFFSET:  DS      01      ;HOLDS ADDRESS CORRECTION CONSTANT
0009                  291     DIR:     DS      01      ;HOLDS DIRECTION FOR SPEED CALCULATED
                      292
                      293              END
```

SYMBOL TABLE LISTING
------ ----- -------

| NAME | TYPE | VALUE | | | ATTRIBUTES |
|---|---|---|---|---|---|
| ACC. . . . | D ADDR | 00E0H | A | | |
| CHKH . . . | C ADDR | 012FH | R | | SEG=USERS |
| CROUT. . . | C ADDR | ---- | | EXT | |
| DIR. . . . | D ADDR | 0009H | R | | SEG=EYEVAR |
| DPL. . . . | D ADDR | 0082H | A | | |
| EOC. . . . | B ADDR | 00B0H.2 | A | | |
| ES1. . . . | C ADDR | 011EH | R | | SEG=USERS |
| ESEND. . . | C ADDR | 013CH | R | | SEG=USERS |
| EX4. . . . | C ADDR | 01B8H | R | | SEG=USERS |
| EXC1 . . . | C ADDR | 0177H | R | | SEG=USERS |
| EXDONE . . | C ADDR | 01B1H | R | | SEG=USERS |
| EXEC . . . | C ADDR | 016BH | R | | SEG=USERS |
| EXECLP . . | C ADDR | 0171H | R | | SEG=USERS |
| EYEVAR . . | D SEG | 000AH | | | REL=UNIT |
| EYSCAN . . | C ADDR | 0115H | R | | SEG=USERS |
| EZON . . . | C ADDR | 01A7H | R | | SEG=USERS |
| HI . . . . | C ADDR | 0196H | R | | SEG=USERS |
| HANDON . . | C ADDR | 009EH | R | | SEG=USERS |
| HEXIN. . . | C ADDR | ---- | | EXT | |
| HGRTR. . . | C ADDR | 0187H | R | | SEG=USERS |
| HIGHON . . | C ADDR | 0134H | R | | SEG=USERS |
| IOBIT. . . | NUMB | 6000H | A | | |
| JTWOON . . | C ADDR | 0032H | R | | SEG=USERS |
| LOWON. . . | C ADDR | 013DH | R | | SEG=USERS |
| MAIN . . . | C ADDR | 0013H | R | | SEG=USERS |
| MES0 . . . | C ADDR | 01DEH | R | | SEG=USERS |
| MES1 . . . | C ADDR | 01D5H | R | | SEG=USERS |
| MES2 . . . | C ADDR | 01EEH | R | | SEG=USERS |
| MES3 . . . | C ADDR | 0200H | R | | SEG=USERS |
| MES4 . . . | C ADDR | 0212H | R | | SEG=USERS |
| MESSAGE . . | C ADDR | 01C7H | R | | SEG=USERS |
| NAM. . . . | NUMB | 0002H | A | PUB | |
| NAM? . . . | C ADDR | ---- | | EXT | |
| NEWLIN . . | C ADDR | ---- | | EXT | |
| OFFSET . . | D ADDR | 0008H | R | | SEG=EYEVAR |
| ONE1 . . . | C ADDR | 003EH | R | | SEG=USERS |
| ONE2 . . . | C ADDR | 0041H | R | | SEG=USERS |
| ONE3 . . . | C ADDR | 0059H | R | | SEG=USERS |
| ONE4 . . . | C ADDR | 005BH | R | | SEG=USERS |
| ONE5 . . . | C ADDR | 006EH | R | | SEG=USERS |
| ONE6 . . . | C ADDR | 0075H | R | | SEG=USERS |
| ONE7 . . . | C ADDR | 0091H | R | | SEG=USERS |
| ONE8 . . . | C ADDR | 009AH | R | | SEG=USERS |
| ONEEXT . . | C ADDR | 0087H | R | | SEG=USERS |
| ONEON. . . | C ADDR | 0035H | R | | SEG=USERS |
| OUTMES . . | D ADDR | ---- | | EXT | |
| P3 . . . . | D ADDR | 00B0H | A | | |
| PAN1 . . . | C ADDR | 015EH | R | | SEG=USERS |

```
NAME     TYPE  VALUE      ATTRIBUTES

RECPOS . . C ADDR  0147H   R       SEG=USERS
RELSB. . . C ADDR  ----      EXT
SCAN . . . C ADDR  00EFH   R       SEG=USERS
SNDPAN . . C ADDR  0158H   R       SEG=USERS
SNDSPD . . C ADDR  00E2H   R       SEG=USERS
SPD1 . . . C ADDR  0194H   R       SEG=USERS
SPDC . . . D ADDR  0004H   R       SEG=EYEVAR
SPDMAL . .   NUMB  1002H   A
SPEED. . . D ADDR  0003H   R       SEG=EYEVAR
SPIN . . . C ADDR  ----      EXT
SPMES. . . C ADDR  ----      EXT
TEMP . . . D ADDR  0000H   R       SEG=EYEVAR
TEMP1. . . D ADDR  0001H   R       SEG=EYEVAR
TEMP2. . . D ADDR  0002H   R       SEG=EYEVAR
TEMPGT . . C ADDR  0084H   R       SEG=USERS
TWO. . . . C ADDR  00BCH   R       SEG=USERS
TW1. . . . C ADDR  00D9H   R       SEG=USERS
TWEXIT . . C ADDR  00DCH   R       SEG=USERS
TWOON. . . C ADDR  00A7H   R       SEG=USERS
TWZON. . . C ADDR  00DFH   R       SEG=USERS
USER . . . C ADDR  0000H   R PUB   SEG=USERS
USERS. . . C SEG   0223H           REL=UNIT
WRMES. . . C ADDR  ----      EXT
ZERO . . . B ADDR  0020H.0 A
ZON. . . . C ADDR  0024H   R       SEG=USERS

REGISTER BANK(S) USED: 0, TARGET MACHINE(S): 8051

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

```
LOC  OBJ             LINE     SOURCE

                        1     $TITLE(USER PROG WITH PAIN AND MEMORY 23-JULY-82)
                        2
                        3     EXTRN CODE (CMD,WRMES,RDMES,SPIN,ECHO,SPMES,CROUT,RELSB,NAM?)
                        4     EXTRN DATA (OUTMES)
                        5     PUBLIC USER,NAM
                        6
                        7     USERS   SEGMENT CODE
----                    8             RSEG    USERS
                        9
  0001                 10     NAM     EQU     01      ;NAME IS 01
                       11
0000 900000   F      12     USER:   MOV     DPTR,#USEMES    ;OUTPUT MESSAGE
0003 120000   F      13             CALL    SPMES
0006 120000   F      14             CALL    ECHO            ;INPUT 'C' OR 'M'
0009 120000   F      15             CALL    RELSB           ;BE SURE TO REL. SBUS
000C B44330          16             CJNE    A,#'C',MEM?
                       17
                       18     ;TWO COMMANDS FOR TEACHING CONDITION RESPONSE TO THE NETWORK
                       19
  00B2                 20     PAIN    BIT     P3.2
  1004                 21     PNMES   EQU     1004H   ;PAIN FLAG TO MOTORS
  1005                 22     EYEVAL  EQU     1005H   ;CURRENT INTENSTY
  1001                 23     VAL1    EQU     1001H   ;HALF OF THE MEMORY LOOP
  1003                 24     VAL3    EQU     1003H   ;THE OTHER HALF
                       25
000F 120000   F      26     COND:   CALL    NAM?            ;ARE YOU BEING CALLED ?
0012 30B2FA          27             JNB     PAIN,COND       ;WAIT FOR PAIN
0015 901005          28     YIPES:  MOV     DPTR,#EYEVAL
0018 120000   F      29             CALL    RDMES           ;READ CURRENT VALUE
001B 120000   F      30     CONDLP: CALL    NAM?
001E 901001          31             MOV     DPTR,#VAL1
0021 120000   F      32             CALL    WRMES           ;START THE MEMORY LOOP
0024 901005          33             MOV     DPTR,#EYEVAL
0027 120000   F      34             CALL    RDMES           ;READ ANOTHER VALUE
002A 7011            35             JNZ     COMP            ;GO COMPARE IF NON-ZERO VALUE
002C 901003          36     CONTU:  MOV     DPTR,#VAL3
002F 120000   F      37             CALL    RDMES           ;GET MEMORY VALUE BACK
0032 120000   F      38             CALL    CROUT
0035 20B2DD          39             JB      PAIN,YIPES      ;LEARN NEW VALUE IF ASKED
0038 120000   F      40             CALL    DLALOP          ;WAIT AWHILE
003B 80DE            41             JMP     CONDLP
                       42
003D 120000   F      43     COMP:   CALL    CROUT
0040 B500E9   F      44             CJNE    A,OUTMES,CONTU  ;IF NOT THE REMEMBERED VALUE CONTINUE
0043 901004          45             MOV     DPTR,#PNMES
0046 E4              46             CLR     A
0047 120000   F      47             CALL    WRMES           ;IF EQUAL SET PAIN FLAG
004A 80E0            48             JMP     CONTU           ;AND CONTINUE
                       49
                       50
```

```
LOC  OBJ            LINE     SOURCE

003C 120000   F       51              CALL    STEP
003F DFF8             52              DJNZ    R7,PAINLP
0041 80CB             53              JMP     EYELOP
                      54
                      55      ;THIS ROUTINE WILL START TIMER WAIT FOR IT TO FINISH
                      56      ;AND OUTPUT THE PULSE FOR PROPER DIRECTION
                      57
0043 758A00           58      STEP:   MOV     TL0,#00
0046 D28C             59              SETB    TR0             ;START TIMER
0048 308DFD           60              JNB     TF0,$           ;WAIT FOR IT TO BE DONE
004B C28D             61              CLR     TF0             ;RESET FLAG
004D C28C             62              CLR     TR0             ;STOP TIMER
004F 300005           63              JNB     DIR.0,CCWL      ;TEST DIR
0052 75B0EB           64              MOV     P3,#CW          ;MOVE CW
0055 8003             65              SJMP    STOPL           ;OR
0057 75B0E7           66      CCWL:   MOV     P3,#CCW         ;MOVE CCW
005A 75B0EF           67      STOPL:  MOV     P3,#STOP        ;OUT REST OF PULSE
005D 22               68              RET
                      69
005E 0A               70      MESAGE: DB 0AH,0DH,'THE MOTOR PROCESSOR IS UP',0AH,0DH,00
005F 0D
0060 54686520
0064 4D6F746F
0068 72205072
006C 6F636573
0070 736F7220
0074 69732055
0078 70
0079 0A
007A 0D
007B 00
                      71
                      72              END
```

```
SYMBOL TABLE LISTING
------ ----- -------

N A M E     T Y P E   V A L U E       A T T R I B U T E S

CMD. . . .  C ADDR    ----      EXT
COMP . . .  C ADDR    003DH   R       SEG=USERS
COND . . .  C ADDR    000FH   R       SEG=USERS
CONDLP . .  C ADDR    001BH   R       SEG=USERS
CONTU. . .  C ADDR    002CH   R       SEG=USERS
CROUT. . .  C ADDR    ----      EXT
DLALOP . .  C ADDR    0068H   R       SEG=USERS
DLALP1 . .  C ADDR    006AH   R       SEG=USERS
ECHO . . .  C ADDR    ----      EXT
EYEVAL . .    NUMB    1005H   A
MEM? . . .  C ADDR    004CH   R       SEG=USERS
MEMORY . .  C ADDR    004FH   R       SEG=USERS
NAM. . . .    NUMB    0001H   A PUB
NAM? . . .  C ADDR    ----      EXT
OUTMES . .  D ADDR    ----      EXT
P3 . . . .  D ADDR    00B0H   A
PAIN . . .  B ADDR    00B0H.2 A
PNMES. . .    NUMB    1004H   A
RDMES. . .  C ADDR    ----      EXT
RELSB. . .  C ADDR    ----      EXT
SPIN . . .  C ADDR    ----      EXT
SPMES. . .  C ADDR    ----      EXT
USEEXT . .  C ADDR    0071H   R       SEG=USERS
USEMES . .  C ADDR    0074H   R       SEG=USERS
USER . . .  C ADDR    0000H   R PUB   SEG=USERS
USERS. . .  C SEG     00A2H           REL=UNIT
VAL1 . . .    NUMB    1001H   A
VAL3 . . .    NUMB    1003H   A
WRMES. . .  C ADDR    ----      EXT
YTPES. . .  C ADDR    0015H   R       SEG=USERS

REGISTER BANK(S) USED: 0, TARGET MACHINE(S): 8051

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

```
LOC  OBJ             LINE     SOURCE

                        1     $TITLE(USER PROG MOTOR PROCESS 14-JULY-82)
                        2     ;THIS PROCESS TAKES A SPEED AND DIR FROM MAILBOX 0002 (THIS MAILBOX)
                        3     ;IS CALLED SPEED). AND ROTATES THE STEPPING MOTOR AT THAT SPEED AND
                        4     ;DIRECTION. THE SPEED IS A FOUR BIT NUMBER IN THE LOWER NIBBLE AND THE
                        5     ;DIRECTION IS BIT 4 : 1 MEANS CW AND 0 IS CCW.
                        6     ;ALSO IT READS A MAILBOX CALLED PAIN (0004). IF IT IS ZERO "PAIN" HAS
                        7     ;OCCURED AND A REFLEX ACTION IS TAKEN (TOP SPEED FOR 90 DEGREES IN THE
                        8     ;OPPISITE DIRECTION.
                        9
  00EB                 10     CW       EQU     0EBH     ;STEPPER WORD FOR CW MOVEMENT
  00E7                 11     CCW      EQU     0E7H     ;CCW MOVEMENT
  00EF                 12     STOP     EQU     0EFH     ;THE WORD THAT MUST FOLLOW EACH OF ABOVE
  00FF                 13     FREE     EQU     0FFH     ;WORD THAT FREES THE MOTOR
  1002                 14     SPEED    EQU     1002H    ;MAILBOX FOR THE SPEED AND DIRECTION
  1004                 15     PAIN     EQU     1004H    ;MAILBOX FOR PAIN FLAG
  0020                 16     DIR      EQU     20H      ;TEMP STORAGE FOR THE DIRECTION
  0064                 17     HALF     EQU     100      ;NUMBER STEPS FOR 90 DEGREES
  0000                 18     NAM      EQU     00
                       19
                       20     EXTRN CODE (RDMES,WRMES,CROUT,SPMES,RELSB,NAM?)
                       21     EXTRN DATA (OUTMES)
                       22     PUBLIC USER,NAM
                       23
                       24     USERS    SEGMENT CODE
  ----                 25              RSEG    USERS
                       26
0000 900000    F      27     USER:    MOV     DPTR,#MESAGE     ;POINT TO MESSAGE FOR TERMNIAL
0003 120000    F      28              CALL    SPMES            ;OUT THE MESSAGE
0006 120000    F      29              CALL    RELSB
0009 758961           30              MOV     TMOD,#61H        ;16 BIT TIMER 0
000C C28D             31              CLR     TF0              ;CLEAR TIMER DONE FLAG
000E 120000    F      32     EYELOP:  CALL    NAM?
0011 901002           33              MOV     DPTR,#SPEED
0014 120000    F      34              CALL    RDMES            ;READ THE SPEED
0017 C4               35              SWAP    A
0018 F520             36              MOV     DIR,A            ;SAVE THE DIRECTION
001A C2E0             37              CLR     ACC.0            ;AND REMOVE IT FROM SPEED
001C 7005             38              JNZ     EYEMOV           ;MOVE MOTOR IF SPEED NOT ZERO
001E 75B0FF           39              MOV     P3,#FREE         ;ELSE FREE MOTOR TO KEEP DRIVERS COOL
0021 80EB             40              JMP     EYELOP
0023 F58C             41     EYEMOV:  MOV     TH0,A            ;SET TIMER VAULE TO SPEED
0025 120000    F      42              CALL    STEP             ;STEP THE MOTOR
0028 901004           43              MOV     DPTR,#PAIN       ;TEST PAIN FLAG
002B 120000    F      44              CALL    RDMES
002E 70DE             45              JNZ     EYELOP           ;IF NOT CONTINUE
0030 74FF             46              MOV     A,#0FFH
0032 120000    F      47              CALL    WRMES            ;RESET THE FLAG
0035 B200             48              CPL     DIR.0            ;CHANGE DIRECTION
0037 7F64             49              MOV     R7,#HALF                 ;SET COUNT FOR 90 DEGREES
0039 758CF5           50     PAINLP:  MOV     TH0,#0F5H        ;SET TOP SPEED
```

```
LOC  OBJ            LINE     SOURCE

                      51     ;THE OTHER COMMAND
004C B44D22           52     MEM?:   CJNE    A,#'M',USEEXT
004F 901001           53     MEMORY: MOV     DPTR,#VAL1
0052 120000    F      54             CALL    RDMES           ;READ MEMORY VALUE
0055 120000    F      55             CALL    DLALOP          ;WAIT FOR AWHILE
0058 901003           56             MOV     DPTR,#VAL3
005B F500      F      57             MOV     OUTMES,A
005D 120000    F      58             CALL    WRMES           ;OUTPUT MEMORY VALUE
0060 120000    F      59             CALL    CROUT           ;ALSO OUT IT TO TERMNIAL
0063 120000    F      60             CALL    NAM?
0066 80E7             61             JMP     MEMORY
                      62
0068 7EFF             63     DLALOP: MOV     R6,#0FFH        ;DELAY LOOP USED BY ABOVE
006A 7FFF             64     DLALP1: MOV     R7,#0FFH
006C DFFE             65             DJNZ    R7,$
006E DEFA             66             DJNZ    R6,DLALP1
0070 22               67             RET
                      68
0071 020000    F      69     USEEXT: JMP     CMD
                      70
0074 0A               71     USEMES: DB 10,13,'INPUT C FOR CONDITIONING AND M FOR MEMORY- ',00
0075 0D
0076 496E7075
007A 7420●4320
007E 666F7220
0082 636F6E64
0086 6974696F
008A 6E696E67
008E 20616E64
0092 204D2066
0096 6F72206D
009A 656D6F72
009E 792D20
00A1 00
                      72
                      73             END
```

SYMBOL TABLE LISTING

| NAME | TYPE | VALUE | | | ATTRIBUTES |
|---|---|---|---|---|---|
| ACC. . . . | D ADDR | 00E0H | A | | |
| CCW. . . . | NUMB | 00E7H | A | | |
| CCWL . . . | C ADDR | 0057H | R | | SEG=USERS |
| CROUT. . . | C ADDR | ---- | | EXT | |
| CW . . . . | NUMB | 00EBH | A | | |
| DIR. . . . | NUMB | 0020H | A | | |
| EYELOP . . | C ADDR | 000EH | R | | SEG=USERS |
| EYEMOV . . | C ADDR | 0023H | R | | SEG=USERS |
| FREE . . . | NUMB | 00FFH | A | | |
| HALF . . . | NUMB | 0064H | A | | |
| MESAGE . . | C ADDR | 005EH | R | | SEG=USERS |
| NAM. . . . | NUMB | 0000H | A | PUB | |
| NAM? . . . | C ADDR | ---- | | EXT | |
| OUTMES . . | D ADDR | ---- | | EXT | |
| P3 . . . . | D ADDR | 00B0H | A | | |
| PAIN . . . | NUMB | 1004H | A | | |
| PAINLP . . | C ADDR | 0039H | R | | SEG=USERS |
| RDMES. . . | C ADDR | ---- | | EXT | |
| RELSB. . . | C ADDR | ---- | | EXT | |
| SPEED. . . | NUMB | 1002H | A | | |
| SPMES. . . | C ADDR | ---- | | EXT | |
| STEP . . . | C ADDR | 0043H | R | | SEG=USERS |
| STOP . . . | NUMB | 00EFH | A | | |
| STOPL. . . | C ADDR | 005AH | R | | SEG=USERS |
| TF0. . . . | B ADDR | 0088H.5 | A | | |
| TH0. . . . | D ADDR | 008CH | A | | |
| TL0. . . . | D ADDR | 008AH | A | | |
| TMOD . . . | D ADDR | 0089H | A | | |
| TR0. . . . | B ADDR | 0088H.4 | A | | |
| USER . . . | C ADDR | 0000H | R | PUB | SEG=USERS |
| USERS. . . | C SEG | 007CH | | | REL=UNIT |
| WRMES. . . | C ADDR | ---- | | EXT | |

REGISTER BANK(S) USED: 0, TARGET MACHINE(S): 8051

ASSEMBLY COMPLETE, NO ERRORS FOUND

# APPENDIX C

## INSTRUCTION FOR PRINTED CIRCUIT BOARD CONSTRUCTION

This Appendix contains instructions that will allow a student to build new printed circuit microcomputer node boards. It is suggested that all instructions be read before beginning.

Table C.1 contains the list of components required. Figs. C.1 and C.2 contain copies of the printed circuit board mask with symbols that are explained in the instructions. Don't rush, make each solder joint carefully and time will be saved later.

1. Cut out the board using the shear in EE machine shop. Cut the board along the inside of the trace running around the outside of the board. Cut across the bus connector fingers at the inside of the markings. Use the small right angle shear to cut out the notches in the lower corners of the board.

2. Drill all holes with a number 64 drill bit. Wear eye protection and be careful not to press too hard on the delicate bit, since these bits break very easily.

3. Redrill the hole in the middle of the voltage regulator pad with a number 28 drill bit.

4. Redrill the two holes for mounting the microswitch with a number 44 drill bit.

5. Install wire connections in all feed through holes. To do

TABLE C.1

Parts List — Parts List for One Node Board

| ITEMS | # OF ITEMS | ITEM NAME |
|---|---|---|
| 1 | Intel 8751 m1 | Intel 8751 microcomputer |
| 1 | Intel 8 35 1 | Intel 8185 1024 x 8-bit RAM |
| 6 | 74LS245 8-bi6 | 74LS245 8-bit bi-directional buffer |
| 1 | 74LS324 Vol1 | 74LS324 Voltage-controlled oscillator |
| 1 | 74LS15" uac1 | 74LS157 Quad 2 to 1 multiplexers |
| 1 | 74LS38 ad 1 | 74LS38 Quad open collector 2-input NAND gates |
| 2 | 74LS37 ad 2 | 74LS37 Quad 2-input NAND buffers |
| 1 | 74LS1C ip1 | 74LS10 Triple 3-input NAND gates |
| 1 | 74LS04 x 1 | 74LS04 Hex inverters |
| 1 | 7805 P f1 | 7805 Plus five volt, voltage regulator |
| 1 | 3/4" 4" 1 | 3/4" x 3/4" Heat sink |
| 1 | 6-32 " 1 | 6-32 x 1/4" Screw |
| 1 | 6-3 1 | 6-32 Nut |
| 1 | 40- e 1 | 40-pin wirewrap socket |
| 6 | 20-pa d 6 | 20-pin solder socket |
| 1 | 18-pir d 1 | 18-pin solder socket |
| 1 | 16-pin d 1 | 16-pin solder socket |
| 7 | 14-pin e7 | 14-pin solder socket |
| 2 | 10 Kil T2 | 10 Kiloohm Trim pot (P1 and P2) |
| 9 | 1 Kiloc 1/9 | 1 Kiloohm 1/4 watt Resister (R1-R9) |
| 1 | .33 mi ar1 | .33 microfarad capacitor (C2) |
| 1 | .01 mi farl | .01 microfarad capacitor (C3) |
| 1 | 5 picof ad 1 | 5 picofarad copacitor (C4) |
| 12 | .10 or . 2 | .10 or .15 microfarad capacitor (bypass) |
| 1 | Micro Switch1 | Micro Switch (S1) |
| 2 | 2-56 X 1 2" 2 | 2-56 X 1/2" Screw |
| 2 | 2-56 Nut 2 | 2-56 Nut |
| 1 | 34-pin Sc tc1 | 34-pin Scotchflex Header |
| 1 | #64 Drill bi1 | #64 Drill bit |

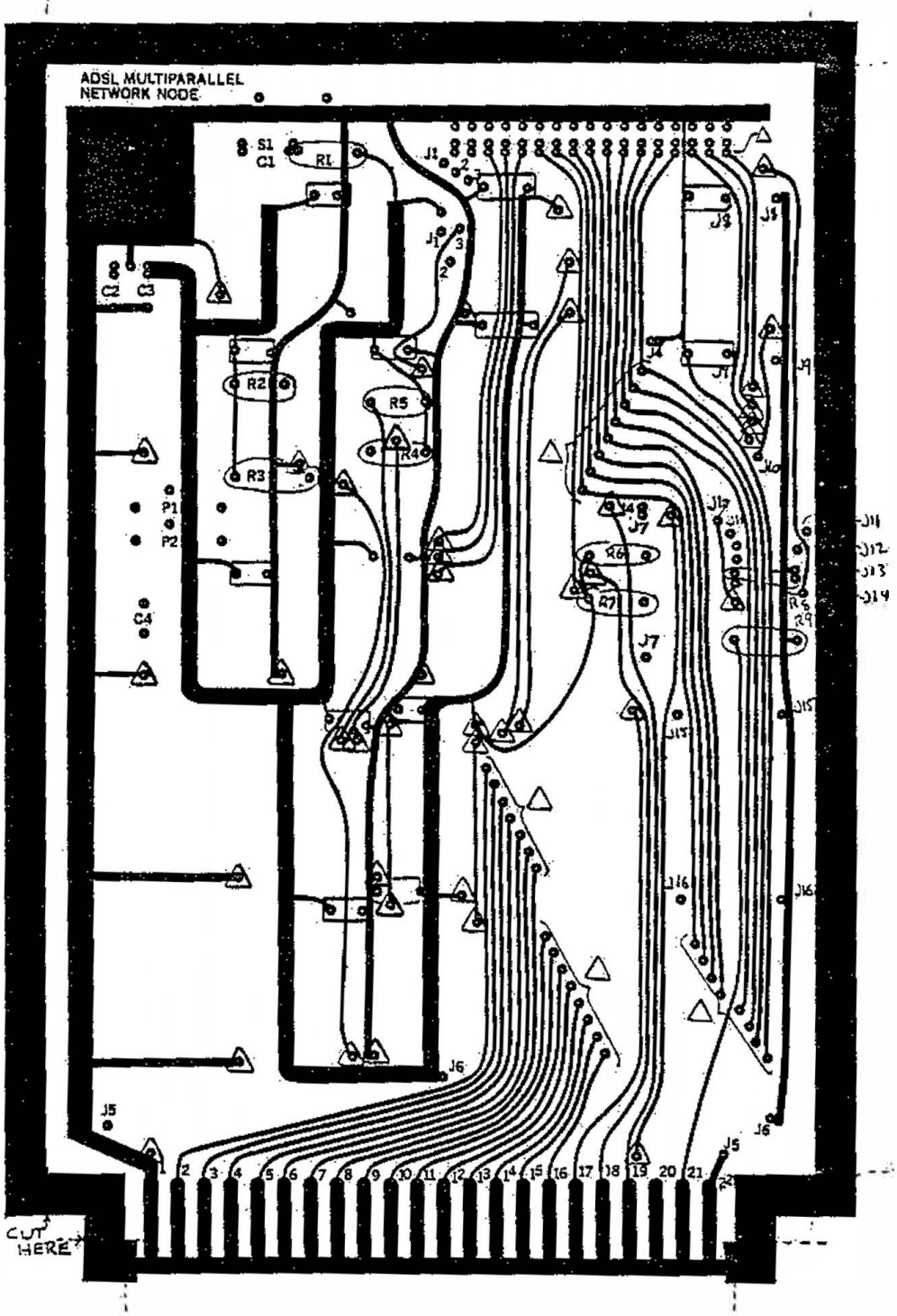



FIGURE C.1

Front side of Printed Circuit Board

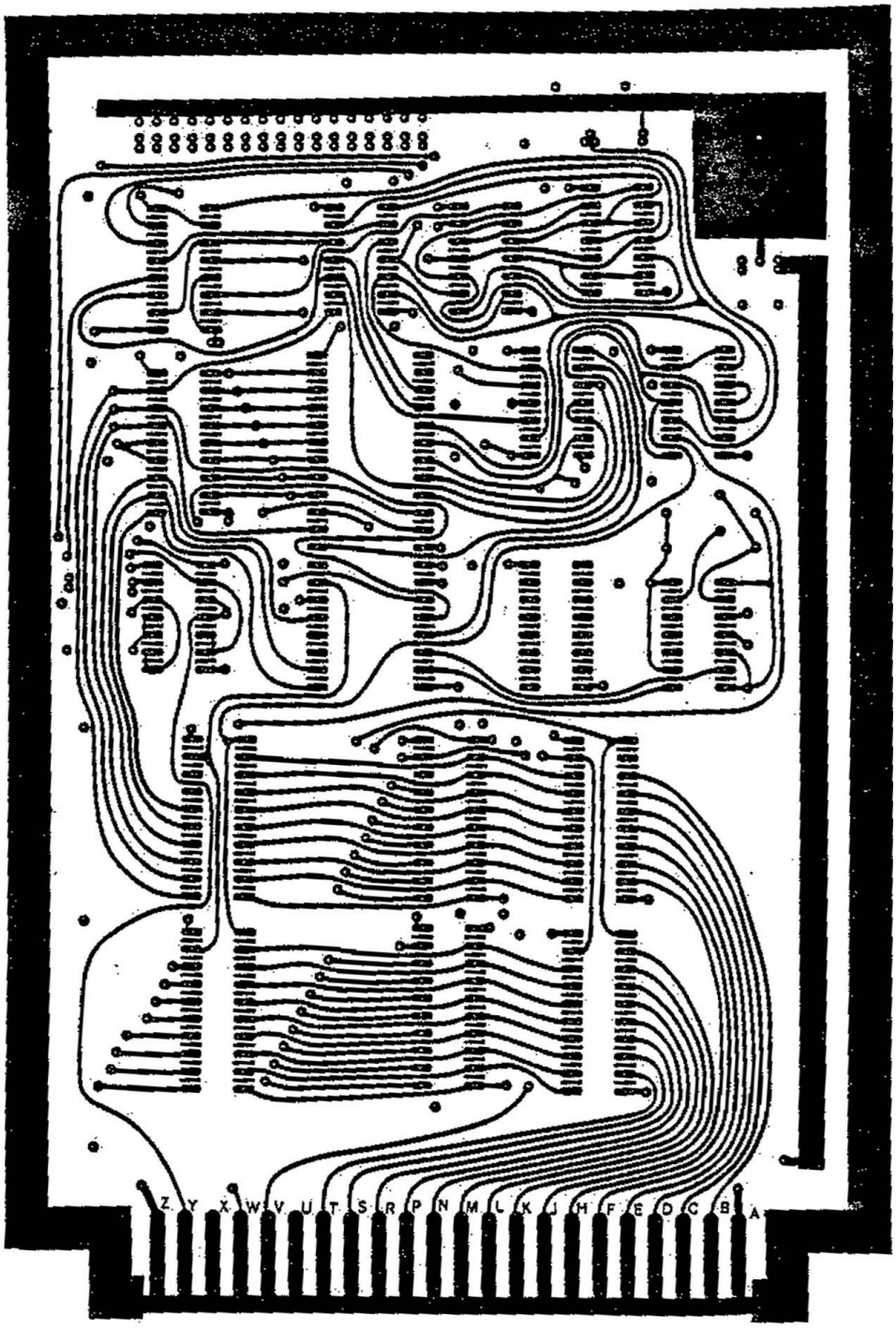



FIGURE C.2

Back Side of Printed Circuit Board

this:

1. Cut many approximately 1/4 inch lenghts of 22 gauge uninsulated wire.

2. Place the board on top of two pieces of 22 gauge insulated wire to hold the board just off the bench.

3. Insert the wire pieces in the feed through holes marked on Figs. C.1 and C.2 by triangles surrounding the pads.

4. Solder the wires on the top side of the board.

5. Trim the wires close to the board.

6. After setting the board on a flat surface with the other side up, solder the wires on this side.

6. Install the sockets, with the exception of the socket for chip A2 which should be installed after the bypass capacitors are installed. The wirewrap socket for the processor should be the last to be soldered in. The C2 socket position should not have a socket placed in it.

7. Install the 34-pin header.

8. Install the .10 or .15 microfarad bypass capacitors. Their pads are marked with squares. These capacitors must be soldered on both sides of the board.

9. Install the resistors (all one Kiloohm). The resistor positions are marked with ovals around the pairs of pads. These too must be soldered on both sides of the board.

10. Install the trim pots, P1 and P2.

11. Install the rest of the capacitors, C2 through C4 (C1 will not be used), soldering on both sides of the board.

12. Install 22 gauge insulated wire as jumpers. Jumpers J1 through J6 are marked on the PC board but the rest are only marked on the diagram. These also must be soldered on both sides of the board.

13. Install the microswitch with the 2-56 hardware and use 22 gauge wire to connect the common and normally open terminals of the switch to the pads.

14. Install the voltage regulator and it heatsink with 6-32 hardware and solder the leads on top and bottom of the board.

15. Double check all work. Each hole should be filled with the

exception of the holes for capacitor C1 and the C2 socket position. Look carefully for unwanted solder bridges all over the board.

16. With an ohm meter, test for any shorts between ground and the input and output of the voltage regulator. There will be a finite resistance becuase of the trimpots. Also test that the upper right pin of each socket is connected to the output of the voltage regulator and the lower left pin of each socket is connected to ground.

17. Place the board in an extender card and place the card in the otherwise empty rack. Be sure that the component side is on your left and bus pins A and 1 are on the top. Turn on the power and check for the plus 5 volts at the output of the voltage regulator and at the power pins of all sockets.

18. Turn off the power, remove the card and install all the chips (see Fig. 3.6 for the layout) including an 8751 with the monitor program in its EPROM.

19. Replace the card in the rack (again being careful that pins A and 1 are at the top) and turn the power on.

20. Adjust the trimpots to get 2 volts at pins 2 and 13 of the 74LS324. Then further adjust the pots to get 7 to 8 megahertz

on the 8751 pin 19. Reset, followed by a CONTROL N and the nodes name, should get the monitor running and the "name:" prompt should appear at the terminal.

21. Use the SX monitor command to test that the XRAM reads and writes can be performed. If so, turn off power, install a working board in a neighboring slot and use the SX command from both monitors to see that the new card can write to the working card and be written to by the working card. If so, the new card is also working. If not, check that the requests are getting in and out of the card and getting to where they are supposed to and that the requests are being granted. Look carefully for solder bridges in and around the buffer sockets.

# APPENDIX D

# INSTRUCTIONS FOR ASSEMBLING AND LINKING USER SOFTWARE AND PROGRAMMING THE 8751'S EPROM

All software for the network is to be developed on the Intel Intellec Development System. This system provides a CRT-based editor, an assembler, a linker, and the EPROM programmer. Below is an example of how a user program called USER is entered, assembled, linked with the monitor software and programmed into the 8751's EPROM.

1. Turn on the Intellec, insert the NETWORK SYSTEM disk, push reset.

2. The 8751 assembly code is entered and edited with a screen editor called CREDIT. After the Intellec's prompt, dash (-), type:

   -CREDIT USER.A51

   The Intellec will respond with:

   ISIS-II CRT-BASED TEXT EDITOR V2.0

   NEW FILE xxx FREE DISK BLOCKS

   The code may now be typed in. Refer to the CREDIT user manual for a detailed description of the editor functions and commands. To exit the editor, hit the HOME key and type EX. The program will have been saved on disk under the filename USER.A51.

3. Now the program can be assembled by typing:

   -ASM51 USER.A51

   The Intellec will respond with:

   ISIS-II MCS-51 MACRO ASSEMBLER, V2.0

ASSEMBLY COMPLETE, xx ERRORS FOUND

The listing will be saved under the name USER.LST. If any errors were found, this file contains the error messages and may be viewed using the command:

-COPY USER.LST TO :CO:

CONTROL S and CONTROL Q stop and start the display so the errors can be noted. To print the listing:

1. Via the lab's arbitrators, connect the Intellec's TTY port to the printer.

2. Set the printer to 9600 baud.

3. Type:

   -PRINT

   ENTER FILENAME will be returned.

   USER.LST should be typed.

   To exit PRINT hit carriage return when asked for filename.

4. To link the user program to the monitor, type:

   -RL51 USER.OBJ,MON.OBJ

   The Intellec will come back with:

   ISIS-II MCS-51 RELOCATOR AND LINKER, V1.0

The linker has named the linked file USER.

5. The code is now ready to be programmed into the 8751's EPROM. To program the 8751's EPROM:

1. Copy the file to the NETWORK UPM disk by typing:

   -COPY USER TO USER

   The following is the response:

   LOAD SOURCE DISK, THEN TYPE (CR)

   The source disk is in, so simply hit carriage return.

   LOAD OUTPUT DISK, THEN TYPE (CR)

   Load the NETWORK UPM disk and hit carriage return.

   LOAD SYSTEM DISK, THEN TYPE (CR)

   Replace the NETWORK SYSTEM disk and hit carriage return.

   The prompt will appear.

2. Replace the NETWORK UPM disk and type:

   -UPM

   The Intellec returns:

   ISIS-II PROM MAPPER

   TYPE*

   Type: 8748

   UPM will return with its prompt, asterisk (*). Type:

   *SOCKET=2

3. Install the UPP-551 socket adapter in socket 2.

4. Turn on the PROM programmer.

5. If the length of the code is not known:

   Type:

   *FILL FROM 0 TO 0FFFH WITH 0FFH

   *READ OBJ FILE USER

   *DISPLAY FROM xxx to yyy

   This will allow the user to search the memory for the end-address of the code (the last non-FFH location). Suggested starting values for the search are xxx=300H and yyy=500H.

6. Place the 8751 in the socket. The notch (pin 1) should be at the top.

7. Type:

   *PROGRAM FROM 0 TO zzz START 0

   Where zzz is the end-address of the code rounded up. For example, if the last location that contains code is 4A8H, zzz should be 500H.

8. The 8751 is now ready to be used in the network.

# REFERENCES

1. Intel Corporation, MCS-51 FAMILY OF SINGLE CHIP MICROCOMPUTERS USER'S MANUAL, January, 1981.

2. National Semiconductor Corporation, DATA ACQUISITION HANDBOOK, 1978.

3. Texas Instruments Incorporated, THE TTY DATA BOOK, Second Edition, 1976.

4. Francisco J. Varela, Department of Anatomy, School of Medicine, University of Colorado, THE ARITHMETIC OF CLOSURE, Chapter 1, III European Meeting on Cybernetics and Systems Research, in Vienna, Austria, April 24, 1976.